A discontinuous Galerkin fast spectral method for multi-species full Boltzmann on streaming multi-processors

Shashank Jaiswal
Purdue University
West Lafayette, Indiana, U.S.A
jaiswal0@purdue.edu

Jingwei Hu
Purdue University
West Lafayette, Indiana, U.S.A
jingweihu@purdue.edu

Julien K. Brillon
Purdue University
West Lafayette, Indiana, U.S.A
jbrillon@purdue.edu

Alina A. Alexeenko
Purdue University
West Lafayette, Indiana, U.S.A
alexeenk@purdue.edu

ABSTRACT

When the molecules of a gaseous system are far apart, say in microscale gas flows where the surface to volume ratio is high and hence the surface forces dominant, the molecule-surface interactions lead to the formation of a local thermodynamically non-equilibrium region extending few mean free paths from the surface. The dynamics of such systems is accurately described by Boltzmann equation. However, the multi-dimensional nature of Boltzmann equation presents a huge computational challenge. With the recent mathematical developments and the advent of petascale, the dynamics of full Boltzmann equation is now tractable. We present an implementation of the recently introduced multi-species discontinuous Galerkin fast spectral (DGFS) method for solving full Boltzmann on streaming multi-processors. The present implementation solves the inhomogeneous Boltzmann equation in span of few minutes, making it at least two order-of-magnitude faster than the present state-of-art stochastic method—direct simulation Monte Carlo—widely used for solving Boltzmann equation. Various performance metrics, such as weak/strong scaling have been presented. A parallel efficiency of 0.96–0.99 is demonstrated on 36 Nvidia Tesla-P100 GPUs.

CCS CONCEPTS

• **Applied computing** → **Mathematics and statistics**; • **Computing methodologies** → *Massively parallel and high-performance simulations.*

KEYWORDS

Multi-species full Boltzmann, Discontinuous Galerkin Fast Spectral, Graphics Processing Units

PASC '19, June 12–14, 2019, Zurich, Switzerland

ACM ISBN 978-1-4503-6770-7/19/06…$15.00
https://doi.org/10.1145/3324989.3325714

ACM Reference Format:
Shashank Jaiswal, Jingwei Hu, Julien K. Brillon, and Alina A. Alexeenko. 2019. A discontinuous Galerkin fast spectral method for multi-species full Boltzmann on streaming multi-processors. In *Proceedings of the Platform for Advanced Scientific Computing Conference (PASC '19), June 12–14, 2019, Zurich, Switzerland*. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3324989.3325714

1 INTRODUCTION

From the fundamental mass/momentum conservation principles, it can be inferred that, in the presence of external forces, say, pressure and temperature gradients, the heavier species moves slower and the lighter species moves faster giving rise to a phenomena termed as diffusion, and effects thereof. Diffusion processes are critical in many applications, for instance, the measurement of the neutrino mass using a windowless gaseous tritium source in the ongoing KATRIN experiment [3]. The dynamics of such systems (and others) are governed by the Boltzmann equation—an integro-differential equation describing the evolution of the distribution function in six-dimensional phase space—which models the dilute gas behavior at the molecular level to accurately describe a wide range of non-continuum flow phenomena, for instance, shocks, expansions into vacuum [11] as well as velocity and thermal slip at gas-solid interfaces [13, 14]. Most rarefied flows of technological interest involve gas mixtures with species diffusion playing a decisive role in turbulent, chemically reacting flows, and evaporation/condensation processes [16].

The approaches for numerical solution of the Boltzmann equation date back to as early as 1940s [5] using, for example, the now widely used direct simulation Monte Carlo (DSMC) method [2]. The DSMC method, based on the kinetic theory of dilute gases, models the binary interactions between particles stochastically. However, it is this stochastic nature, that makes the method unsuitable for flows involving species in trace concentration, for instance, to analyze the spectrum of beta electrons emitted by tritium source which can be substantially different in the presence of the impurities in KATRIN experiment [3, 15].

The main difficulty of numerically solving the full Boltzmann equation lies in its complicated collision term. Recently, a fast Fourier spectral method for the multi-species Boltzmann collision

operator was introduced in [7]. The complexity for a single evaluation of the collision operator is reduced from $O(N^6)$ (direct calculation) to $O(MN_\rho N^3 \log N)$ (based on a low-rank decomposition strategy), where N is the number of discretization points in each velocity dimension, $N_\rho \sim O(N)$ is the number of discretization points in the radial direction needed for low-rank decomposition, and $M \ll N^2$ is the number of discretization points on the sphere. Based on [6], a discontinuous Galerkin fast spectral (DGFS) method was also proposed in [7] for solving the full multi-species Boltzmann equation. DGFS can produce high order spatially and temporally accurate solutions for low-speed and unsteady flows in micro-systems, and is amenable to excellent nearly-linear scaling characteristics on massively parallel architectures. This paper focuses on implementation aspects of multi-species DGFS with an emphasis on establishing the algorithmic behavior of such numerical schemes. More specifically, here, we are concerned about the scaling characteristics of DGFS on multi-GPU/multi-CPU systems.

In the section that follows, we describe the multi-species Boltzmann in brief, followed by a description of the collision operator algorithm. Various performance metrics such as weak/strong scaling, and micro benchmarks involving, static adaptivity of cell-size and polynomial order approximation for rarefied gas-flows have been discussed in section 3. Concluding remarks are given in section 4.

2 THE MULTI-SPECIES BOLTZMANN EQUATION

The *non-dimensional* Boltzmann equation for multi-species, monoatomic gas without external forces can be written as (cf. [7])

$$\frac{\partial f^{(p)}}{\partial t} + \mathbf{c}\cdot\nabla_{\hat{\mathbf{x}}} f^{(p)} = \sum_q \frac{1}{\mathrm{Kn}_{pq}} Q^{(pq)}, \quad p = 1, 2, \ldots, n, \tag{1}$$

where n denotes number of species in the mixture – each of them represented by a number distribution function $f^{(p)}(t, \mathbf{x}, \mathbf{c})$ of time t, position $\mathbf{x}$, and particle velocity $\mathbf{c}$. The collision operator $Q^{(pq)}$ takes into account interactions between species p and q, which acts only in the velocity space:

$$Q^{(pq)}(f^{(p)}, f^{(q)})(\mathbf{c}) = \int_{\mathbb{R}^3}\int_{S^2} B_{pq}(|\mathbf{c}-\mathbf{c}_*|, \boldsymbol{\sigma}\cdot\widehat{(\mathbf{c}-\mathbf{c}_*)}) \left[f^{(p)}(\mathbf{c}')f^{(q)}(\mathbf{c}'_*) - f^{(p)}(\mathbf{c})f^{(q)}(\mathbf{c}_*)\right] \mathrm{d}\boldsymbol{\sigma}\,\mathrm{d}\mathbf{c}_*, \tag{2}$$

where $(\mathbf{c}, \mathbf{c}_*)$ and $(\mathbf{c}', \mathbf{c}'_*)$ denote the pre and post collision velocity pairs, which are related through momentum and energy conservation as

$$\begin{cases} \mathbf{c}' = \dfrac{\mathbf{c}+\mathbf{c}_*}{2} + \dfrac{(1-m_q/m_p)}{2(1+m_q/m_p)}(\mathbf{c}-\mathbf{c}_*) + \dfrac{1}{(m_p/m_q+1)}|\mathbf{c}-\mathbf{c}_*|\boldsymbol{\sigma}, \\ \mathbf{c}'_* = \dfrac{\mathbf{c}+\mathbf{c}_*}{2} + \dfrac{(1-m_q/m_p)}{2(1+m_q/m_p)}(\mathbf{c}-\mathbf{c}_*) - \dfrac{1}{(m_q/m_p+1)}|\mathbf{c}-\mathbf{c}_*|\boldsymbol{\sigma}, \end{cases} \tag{3}$$

where m_p, m_q denote the mass of particles of species p and q respectively. Here, the vector $\boldsymbol{\sigma}$ varies over the unit sphere $\mathcal{S}^2$. The quantity $\mathcal{B}_{pq}$ (≥ 0) is the collision kernel depending only on $|\mathbf{c}-\mathbf{c}_*|$ and the scattering angle χ (angle between $\mathbf{c}-\mathbf{c}_*$ and $\mathbf{c}'-\mathbf{c}'_*$). In the present work, we consider the variable soft sphere (VSS) [9] scattering model. It is worth emphasizing that although the VSS collision kernel is adopted in the present work for easy comparison with DSMC solutions, the fast spectral method we use for the collision operator applies straightforwardly to general collision kernels (see [4, 6, 7]).

For Variable Soft-Sphere model [2] in particular, the non-dimensional collision-kernel $B_{(pq)}$, and the Knudsen number Kn_{pq} are given as

$$B_{(pq)} = \frac{1}{\sqrt{1+m_p/m_q}} \frac{1}{\left(\frac{m_p m_q}{m_p+m_q}\right)^{(\omega_{pq}-0.5)}} \frac{\alpha_{pq}}{2^{1+\alpha_{pq}}\,\Gamma(2.5-\omega_{pq})\pi} |\mathbf{c}-\mathbf{c}_*|^{2(1-\omega_{pq})} (1+\cos\chi)^{\alpha_{pq}-1}, \tag{4}$$

$$\mathrm{Kn}_{pq} = \frac{1}{\sqrt{1+m_p/m_q}\;\pi\; n_0\; d^2_{(\mathrm{ref},pq)}\; (T_{\mathrm{ref},pq}/T_0)^{\omega_{pq}-0.5}\; H_0}. \tag{5}$$

Here Γ denotes the usual Gamma function, $d_{(\mathrm{ref},pq)}$, $T_{(\mathrm{ref},pq)}$, ω_{pq}, and α_{pq} are, respectively, the reference diameter, the reference temperature, the viscosity index, and the scattering parameter. The diameter $d_{(\mathrm{ref},pq)}$ and exponent α_{pq} are determined so that the transport (viscosity and diffusion) coefficients of VSS are consistent with experimental data. Additionally H_0, T_0, n_0, and m_0, respectively, denote the characteristic length, characteristic temperature, characteristic number density, and characteristic mass m_0. Based upon these, we define the characteristic velocity as $u_0 = \sqrt{2k_B T_0/m_0}$ where k_B refers to Boltzmann constant; and characteristic time as $t_0 = H_0/u_0$. For convenience, we define a pre-factor $\beta^{(pq)}$ as

$$\beta^{(pq)} = \frac{1}{\mathrm{Kn}_{pq}} \frac{1}{\sqrt{1+m_p/m_q}} \frac{1}{\left(\frac{m_p m_q}{m_p+m_q}\right)^{(\omega_{pq}-0.5)}} \frac{\alpha_{pq}}{2^{1+\alpha_{pq}}\,\Gamma(2.5-\omega_{pq})\pi} \tag{6}$$

Henceforth, we will always refer to the non-dimensional Boltzmann equation (1) in our presentation.

2.1 The collision operator

First, note that $Q^{(pq)}(f^{(p)}, f^{(q)})$ does not depend on spatial coordinate $\mathbf{x}$. Given distribution functions $f^{(p)}$ and $f^{(q)}$ of species p and q, dependent only on the velocity coordinate $\mathbf{c}$: discretized *uniformly* using N^3 points, the method produces $Q^{(pq)}(f^{(p)}, f^{(q)})$ at the same grid with $O(MN_\rho N^3 \log N)$ complexity, where $N_\rho \sim O(N)$ is the number of Gauss-Legendre quadrature/discretization points in the radial direction needed for low-rank decomposition , $M \ll N^2$ is the number of discretization points on the sphere. The steps (based on [7]) for evaluating $Q^{(pq)}$ can be summarized as:

- Change the variable $\mathbf{c}_*$ to $\mathbf{u} = \mathbf{c} - \mathbf{c}_*$:

$$Q^{(pq)}(f^{(p)}, f^{(q)})(\mathbf{c}) = \int_{\mathbb{R}^3}\int_{\mathcal{S}^2} B_{pq}(|\mathbf{u}|, \boldsymbol{\sigma}\cdot\hat{\mathbf{u}}) \left[f^{(p)}(\mathbf{c}')f^{(q)}(\mathbf{c}'_*) - f^{(p)}(\mathbf{c})f^{(q)}(\mathbf{c}-\mathbf{u})\right] \mathrm{d}\boldsymbol{\sigma}\mathrm{d}\mathbf{u}, \tag{7}$$

where $\hat{\mathbf{u}}$ is the unit vector along $\mathbf{u}$, and

$$\begin{cases} \mathbf{c}' = \mathbf{c} - \dfrac{m_q}{m_p+m_q}\mathbf{u} + \dfrac{m_q}{m_p+m_q}|\mathbf{u}|\boldsymbol{\sigma}, \\ \mathbf{c}'_* = \mathbf{c} - \dfrac{m_q}{m_p+m_q}\mathbf{u} - \dfrac{m_p}{m_p+m_q}|\mathbf{u}|\boldsymbol{\sigma}. \end{cases} \tag{8}$$

- Determine the extent of velocity domain $D_L = [-L, L]^3$, and periodically extend f, g to $\mathbb{R}^3$.
- Truncate the integral in $\mathbf{u}$ to a ball B_R with

$$R = \frac{4}{1 + \max(4m_q/(m_p+m_q), 2) + \sqrt{1 + m_q/m_p}} L \tag{9}$$

- Approximate $f^{(p)}$, $f^{(q)}$ by truncated Fourier series

$$f^{(p)}(\mathbf{c}) = \sum_{k=-N/2}^{N/2-1} \hat{f}_k^{(p)} e^{i\frac{\pi}{L}k\cdot\mathbf{c}}, \quad f^{(q)}(\mathbf{c}) = \sum_{k=-N/2}^{N/2-1} \hat{f}_k^{(q)} e^{i\frac{\pi}{L}k\cdot\mathbf{c}}. \tag{10}$$

  Note here k is a three-dimensional index.
- Substitute $f^{(p)}$, $f^{(q)}$ into (7), and perform the standard Galerkin projection

$$\begin{aligned} \hat{Q}_k^{(pq)} &:= \frac{1}{(2L)^3}\int_{D_L} Q^{(pq)}(f^{(p)}, f^{(q)})(\mathbf{c}) e^{-i\frac{\pi}{L}k\cdot\mathbf{c}} d\mathbf{c} \\ &= \sum_{\substack{l,m=-N/2 \\ l+m=k}}^{N/2-1} \left[G^{(pq)+}(l,m) - G^{(pq)-}(m,m)\right] \hat{f}_l^{(p)} \hat{f}_m^{(q)}, \end{aligned} \tag{11}$$

  where $k = -N/2, \ldots, N/2-1$, and the kernel modes $G^{(pq)+}$ and $G^{(pq)-}$ are given by

$$\begin{aligned} G^{(pq)+}(l,m) &= \int_{\mathcal{B}_R}\int_{S^2} B_{pq}(|\mathbf{u}|, \boldsymbol{\sigma}\cdot\hat{\mathbf{u}}) \\ &\qquad \left[e^{-i\frac{\pi}{L}\frac{m_q}{m_p+m_q}(l+m)\cdot\mathbf{u} + i\frac{\pi}{L}|\mathbf{u}|\left(\frac{m_q}{m_p+m_q}l - \frac{m_p}{m_p+m_q}m\right)\cdot\boldsymbol{\sigma}}\right] d\boldsymbol{\sigma}\, d\mathbf{u} \\ G^{(pq)-}(m,m) &= \int_{\mathcal{B}_R}\int_{S^2} B_{pq}(|\mathbf{u}|, \boldsymbol{\sigma}\cdot\hat{\mathbf{u}}) \left[e^{-i\frac{\pi}{L}m\cdot\mathbf{u}}\right] d\boldsymbol{\sigma}\, d\mathbf{u}. \end{aligned} \tag{12}$$

It is clear that the direct evaluation of $\hat{Q}_k^{(pq)}$ (for all k) would require $O(N^6)$ complexity. But if we can find a low-rank, separated expansion of $G^{(pq)+}(l,m)$ as

$$G^{(pq)+}(l,m) \approx \sum_{r=1}^{N_\rho} \alpha_r(l+m)\, \beta_r(l)\, \gamma_r(m), \tag{13}$$

then the gain term (positive part) of $\hat{Q}_k^{(pq)}$ can be rearranged as

$$\hat{Q}_k^{(pq)+} = \sum_{r=1}^{N_\rho} \alpha_r(k) \sum_{\substack{l,\ m=-N/2 \\ l+m=k}}^{N/2-1} \left(\beta_r(l)\hat{f}_l^{(p)}\right)\left(\gamma_r(m)\hat{f}_m^{(q)}\right), \tag{14}$$

which is a convolution of two functions $\beta_r(l)\hat{f}_l^{(p)}$ and $\gamma_r(m)\hat{f}_m^{(q)}$, hence can be computed via fast Fourier transform (FFT) in $O(N_\rho N^3 \log N)$ operations. Note that the loss term (negative part) of $\hat{Q}_k^{(pq)-}$ is readily a convolution and can be computed via FFT in $O(N^3 \log N)$ operations.

In order to find the approximation in (13), we simplify (12) as

$$\begin{aligned} G^{(pq)+}(l,m) &= \int_{\mathcal{B}_R}\int_{S^2} B_{pq}(|\mathbf{u}|, \boldsymbol{\sigma}\cdot\hat{\mathbf{u}}) \\ &\qquad e^{-i\frac{\pi}{L}\frac{m_q}{m_p+m_q}(l+m)\cdot\mathbf{u} + i\frac{\pi}{L}|\mathbf{u}|\left(\frac{m_q}{m_p+m_q}l - \frac{m_p}{m_p+m_q}m\right)\cdot\boldsymbol{\sigma}} d\boldsymbol{\sigma}\, d\mathbf{u} \\ &= \int_0^R \int_{S^{d-1}} F^{(pq)}(l+m, \rho, \boldsymbol{\sigma}) e^{i\frac{\pi}{L}\rho\left(\frac{m_q}{m_p+m_q}l - \frac{m_p}{m_p+m_q}m\right)\cdot\boldsymbol{\sigma}} d\boldsymbol{\sigma}\, d\rho, \end{aligned} \tag{15}$$

where

$$F^{(pq)}(l+m, \rho, \boldsymbol{\sigma}) = \rho^2 \int_{S^2} B_{pq}(\rho, \boldsymbol{\sigma}\cdot\hat{\mathbf{u}}) e^{-i\frac{\pi}{L}\rho\frac{m_q}{m_p+m_q}(l+m)\cdot\hat{\mathbf{u}}} d\hat{\mathbf{u}}, \tag{16}$$

while for the loss term,

$$\begin{aligned} G^{(pq)-}(m) &= \int_{\mathcal{B}_R}\int_{S^2} B_{pq}(|\mathbf{u}|, \boldsymbol{\sigma}\cdot\hat{\mathbf{u}})\, e^{-i\frac{\pi}{L}m\cdot\mathbf{u}}\, d\boldsymbol{\sigma}\, d\mathbf{u} \\ &= \int_0^R \int_{S^2}\int_{S^2} \rho^2\, B_{pq}(\rho, \boldsymbol{\sigma}\cdot\hat{\mathbf{u}})\, e^{-i\frac{\pi}{L}\rho\, m\cdot\hat{\mathbf{u}}}\, d\boldsymbol{\sigma}\, d\hat{\mathbf{u}}\, d\rho. \end{aligned} \tag{17}$$

The numerical error in the Fourier-spectral approximation is dependent on several factors: a) the truncation of the infinitely-long velocity mesh to a finite $[-L, L]^3$ interval, b) the size $[-L, L]^3$ of the finite velocity-mesh, c) number of points in the velocity-mesh N, d) choice of selected Gaussian quadrature, e) choice of spherical quadrature, f) finite precision round-off errors, etc. For mathematical details on the error, the reader is referred to [4, 7, 12]. This discussion has been omitted in the present work for brevity.

2.2 The collision operator algorithm

The collision operator procedure described above is applicable for general collision kernels for n-species mixture. However, for a concise description of the algorithmic ideas from an implementation viewpoint, we restrict our discussion to Variable Soft Sphere collision kernel (4). The ideas, however, can certainly be carried over to other collision kernels.

In multi-species implementation, with the high amount of involved computation, our motive is to avoid spurious computation for every timestep. We first outline the procedure for pre-computing variables that can be stored and reused during the course of the simulation.

- First, we precompute $(\pi/L\ \rho\ l\cdot\boldsymbol{\sigma})$. We use Gauss-Legendre-Quadrature (GLQ) for integration. So ρ, the GLQ zeros, is an array of size N_ρ (since the integrand oscillates on the scale of O(N), the total number of quadrature points needed should be $\sim O(N)$). Additionally, we use *spherical design* [19] quadrature on sphere. So, $\boldsymbol{\sigma}$, the spherical-quadrature zeros, is an array of size M. l as previously defined is the 3-D velocity-space index, and is therefore an array of size N^3. Based upon these $(\pi/L\ \rho\ l\cdot\boldsymbol{\sigma})$ is precomputed and stored as a $N_\rho \times M \times N^3$ flattened row-major array a_{xyz}. This is described in steps 1–9 of Algo. (1).
- Second, we compute $F(l+m, \rho, \boldsymbol{\sigma})$ as per Eq. (16). Note that $k = l+m$ is velocity-space index of size N^3. Since $l+m$, ρ, and $\boldsymbol{\sigma}$ do not change with time, the term $F(l+m, \rho, \boldsymbol{\sigma})$ is precomputed and stored as a $N_\rho \times M \times N^3$ flattened row-major

array $\mathrm{b}_{xyz}^{(pq)}$ for every collision pair (p, q). This is described in step 13 of Algo. (1).
- Third, we perform precomputation needed for loss-term $G^{(pq)-}(m)$ as per Eq. (17). The output is stored as a N^3 flattened row-major array $\mathrm{c}_z^{(pq)}$ for every collision pair (p, q). This is described in step 14 of Algo. (1).

Algorithm 1: Pre-computation for Collision-Algorithm

Input: Number of points in each-direction of velocity mesh N, number of quadrature points for low-rank decomposition N_ρ, number of points on half-sphere M, number of points on pre-computation sphere $M^{(\mathrm{pre})}$, spherical quadrature weight $w_{\boldsymbol{\sigma}}$, spherical quadrature-points $\boldsymbol{\sigma}$ (vector-field size: M), pre-computation spherical quadrature weight $w_{\boldsymbol{\sigma}}^{(\mathrm{pre})}$, pre-computation spherical quadrature-points $\boldsymbol{\sigma}^{(\mathrm{pre})}$ (vector-field size: $M^{(\mathrm{pre})}$), Gauss quadrature-weights w_ρ (size: N_ρ), Gauss quadrature-points ρ (size: N_ρ), first collision parameter $\gamma_{pq} = 2(\omega_{pq} - 1)$, second collision parameter $\eta_{pq} = (\alpha_{pq} - 1)$, size of velocity mesh L, normalized mass m_p, m_q of species-pair (p, q)

Output: a,b,c

Declare:
a (size: $MN_\rho N^3$), $\mathrm{b}^{(pq)}$ (size: $MN_\rho N^3$), $\mathrm{c}^{(pq)}$ (size: N^3)
l (vector-field size: N^3), v (size: N)

for $x = 0$ to $N - 1$ **do**
  $\mathrm{v}_x = x$ - $(x \geq \mathrm{N}/2) \times \mathrm{N}$
end for
`// See octave function: [lx,ly,lz]=ndgrid(v)`
$l \leftarrow$ `ndgrid(v)`
`// Subscript x,y,z on symbols denote array-index`
for $x = 1$ to N_ρ **do**
  for $y = 1$ to M **do**
    for $z = 1$ to N^3 **do**
      $\mathrm{a}_{xyz} \leftarrow \pi/\mathrm{L} \times \rho_x \times (l_z \cdot \boldsymbol{\sigma}_y)$
      `// ( · ) denotes vector dot-product`
    end for
    for $\hat{y} = 1$ to $M^{(\mathrm{pre})}$ **do**
      $\mathrm{B}_{pq} \leftarrow \left(1 + \boldsymbol{\sigma}_y \cdot \boldsymbol{\sigma}_{\hat{y}}^{(\mathrm{pre})}\right)^{\eta_{pq}}$
      for $z = 1$ to N^3 **do**
        $\mathrm{b}_{xyz}^{(pq)} \leftarrow \mathrm{b}_{xyz}^{(pq)} + \mathrm{B}_{pq} \times w_{\boldsymbol{\sigma}}^{(\mathrm{pre})} \times \rho_x^{\gamma_{pq}+2} \times \exp(\mathtt{-1i} \times m_q/(m_p + m_q) \times \pi/\mathrm{L} \times \rho_x \times (l_z \cdot \boldsymbol{\sigma}_{\hat{y}}^{(\mathrm{pre})}))$
        $\mathrm{c}_z^{(pq)} \leftarrow \mathrm{c}_z^{(pq)} + (w_\rho)_x \times w_{\boldsymbol{\sigma}} \times \mathrm{B}_{pq} \times w_{\boldsymbol{\sigma}}^{(\mathrm{pre})} \times \rho_x^{\gamma_{pq}+2} \times \exp(\mathtt{-1i} \times \pi/\mathrm{L} \times \rho_x \times (l_z \cdot \boldsymbol{\sigma}_{\hat{y}}^{(\mathrm{pre})}))$
        `// The variables` $\mathrm{b}_{xyz}^{(pq)}$, $\mathrm{c}_z^{(pq)}$ `needs to be computed for every` (p, q) `collision pair`
      end for
    end for
  end for
end for
return a,$\mathrm{b}^{(pq)}$,$\mathrm{c}^{(pq)}$

Next we outline the procedure for computing $Q^{(pq)}$. Recall that our motive is to compute (11)

- First, we compute the forward Fourier transform of $\mathcal{F}_{i,\,l_1}^{(p)}$, and $\mathcal{F}_{i,\,l_2}^{(q)}$ to obtain $\hat{f}_l^{(p)}$ and $\hat{f}_m^{(q)}$ respectively. This is described in step 1 of Algo. (2).
- Second, we compute $G^{(pq)+}(l, m)$ as per Eq. (15). Recall that $(\pi/L\ \rho\ l \cdot \boldsymbol{\sigma})$ has been already precomputed and stored as a_{xyz}. Also recall that $F(l + m, \rho, \boldsymbol{\sigma})$ has been precomputed and stored as $\mathrm{b}_{xyz}^{(pq)}$. These can be reused to compute $G(l, m)$. This is described in step 2–8 of Algo. (2). In our implementation, we explicitly unroll the nested loops using Mako [1] templating engine, such that variables `t1`, `t2` in steps 4 and 5 are computed in a single kernel call (thereby requiring a space of $MN_\rho N^3$ each), and the FFT transforms in the step 6 are rather MN_ρ batched FFT transforms, each of size N^3.
- Third, in order to perform convolution for the loss-term $G^{(pq)-}(l, m)$, we prepare the variable `QG` in step 7 of Algo. (2).
- Fourth, we perform convolutions to compute $\hat{Q}_k^{(pq)}$ as in Eq. (11). Recall that $G^{(pq)-}(m)$ has now been precomputed and stored as `QG`, and can be reused here. An inverse Fourier transform is then performed to obtain final $Q^{(pq)}$. This is described in step 10 of Algo. (2).

3 MICRO-BENCHMARKS

Verification for standard rarefied gas flows can be found in [7]. In the present work, we focus on the evaluation of the algorithmic behavior.

3.1 Hardware Configuration

Serial and parallel implementations of multi-species DGFS solver are run on 15-node Brown-GPU RCAC cluster at Purdue University. Each node is equipped with two 12-core Intel Xeon Gold 6126 CPU, and three Tesla-P100 GPU. The operating system used is 64-bit CentOS 7.4.1708 (Core) with NVIDIA Tesla-P100 GPU accompanying CUDA driver 8.0 and CUDA runtime 8.0. The GPU has 10752 CUDA cores, 16GB device memory, and compute capability of 6.0. The solver has been written in Python/PyCUDA and is compiled using OpenMPI 2.1.0, g++ 5.2.0, and nvcc 8.0.61 compiler with third level optimization flag. All the simulations are done with double precision floating point values. The source-code of the implementation can be found at https://github.com/jaisw7/dgfs1D_gpu.

3.2 Spatially homogeneous case: Krook-Wu exact solution

For constant collision kernel, an exact solution to the spatially homogeneous multi-species Boltzmann equation can be constructed (see [10]). We use this solution to verify the accuracy of the proposed fast spectral method for approximating the collision operator. Considering a binary mixture ($n = 2$; $p = 1, 2$), the equation simplifies to

$$\partial_t f^{(p)} = \sum_{q=1}^{2} \int_{\mathbb{R}^3} \int_{S^2} B_{pq} \left[f^{(p)}(v') f^{(q)}(v'_*) - f^{(p)}(v) f^{(q)}(v_*) \right] \mathrm{d}\boldsymbol{\sigma}\, \mathrm{d}v_*, \tag{18}$$

Algorithm 2: Collision-Algorithm Pseudo-code

Input: Number of points in each-direction of velocity mesh N, Distribution-functions $\mathcal{F}^{(p)}_{i,\,l_1}$ and $\mathcal{F}^{(q)}_{i,\,l_2}$ (size: N^3), number of points on half-sphere M, spherical quadrature weight $w_{\boldsymbol{\sigma}}$, Gauss quadrature-weights w_ρ (size: N_ρ), precomputed variable a (size: $MN_\rho \times N^3$), precomputed variable $\mathrm{b}^{(pq)}$ (size: $MN_\rho \times N^3$), precomputed variable $\mathrm{c}^{(pq)}$ (size: N^3), the kernel prefactor $\beta^{(pq)}$, normalized mass m_p, m_q of species-pair (p, q)

Output: Q

Declare:
{t1,. . . ,t3} (each size: N^3); Q, QG (each size: N^3)

Compute forward FFT:
FTf $\leftarrow$ `fft`$(\mathcal{F}^{(p)}_{i,\,l_1})$
FTg $\leftarrow$ `fft`$(\mathcal{F}^{(q)}_{i,\,l_2})$
// Subscript x,y on symbols denote array-index
// Inner-most loop $r \in \{1, \ldots, N^3\}$ has been ignored
for $x = 1$ to N_ρ **do**
  for $y = 1$ to M **do**
    t1 $\leftarrow \exp(\mathtt{1i} \times m_q/(m_p + m_q) \times \mathrm{a}_{xy}) \times$ FTf
    // Note: These are array-operations over N^3 (z index)
    // 1i denotes the complex number $\sqrt{-1}$
    t2 $\leftarrow \exp(\mathtt{-1i} \times m_p/(m_q + m_p) \times \mathrm{a}_{xy}) \times$ FTg
    // ifft denotes inverse FFT
    t3 $\leftarrow$ `fft`(`ifft`(t1) $\times$ `ifft`(t2))
    QG $\leftarrow$ QG + $(w_\rho)_x \times w_{\boldsymbol{\sigma}} \times \mathrm{b}^{(pq)}_{xy} \times$ t3
  end for
end for
// real returns real part of complex number
Q = $\beta^{(pq)} \times$ `real`(`ifft`(QGs) - $\mathcal{F}^{(p)}_{i,\,l_1} \times$ `ifft`$(\mathrm{c}^{(pq)} \times$ FTg))
return Q

where $B_{pq} = B_{qp} := \frac{\lambda_{qp}}{4\pi n^{(q)}}$ and λ_{pq} is some positive constant. The exact solution is given by

$$f^{(p)}(t, v) = n^{(p)}\left(\frac{m_p}{2\pi K}\right)^{3/2} \exp\left(-\frac{m_p v^2}{2K}\right)\left((1 - 3Q_p) + \frac{m_p}{K} Q_p v^2\right), \tag{19}$$

where

$$\mu = \frac{4m_1 m_2}{(m_1 + m_2)^2}, \quad \tau_1 = \lambda_{22} - \lambda_{21}\mu(3 - 2\mu), \quad \tau_2 = \lambda_{11} - \lambda_{12}\mu(3 - 2\mu),$$
$$A = \frac{1}{6}\left(\lambda_{11} + \lambda_{21}\mu\left(3 - 2\mu\frac{\tau_2}{\tau_1}\right)\right), \quad B = \frac{1}{3}\left(\lambda_{11}\tau_1 + \lambda_{21}\mu(3 - 2\mu)\tau_2\right),$$
$$Q(t) = \frac{A}{A\exp(At) - B}, \quad Q_p(t) = \tau_p Q(t),$$
$$K(t) = \frac{n^{(1)} + n^{(2)}}{(n^{(1)} + n^{(2)}) + 2(n^{(1)}\tau_1 + n^{(2)}\tau_2)Q(t)}. \tag{20}$$

Furthermore, the following condition needs to be satisfied

$$(\tau_1 - \tau_2)\left(2\mu^2\left(\frac{\lambda_{21}}{\tau_1} - \frac{\lambda_{12}}{\tau_2}\right) - 1\right) = 0. \tag{21}$$

For simplicity, we choose $n^{(1)} = n^{(2)} = 1$, $\lambda_{11} = \lambda_{22} = 1$, $\lambda_{12} = \lambda_{21} = 1/2$ but vary the mass ratio m_1/m_2 in the following tests.

It is also helpful to take the derivative of eqn. (19), which yields

$$\partial_t f^{(p)} = f^{(p)}\left(-\frac{3}{2K}K' + \frac{m_p\, v^2}{2K^2}K'\right)$$
$$+ n^{(p)}\left(\frac{m_p}{2\pi K}\right)^{3/2} \exp\left(-\frac{m_p v^2}{2K}\right)\left(-3Q_p' + \frac{m_p}{K}Q_p' v^2 - \frac{m_p}{K^2}K' Q_p v^2\right)$$
$$:= \sum_{q=1}^{2} \boldsymbol{Q}^{(pq)}(f^p, f^q), \tag{22}$$

where

$$Q'(t) = -\frac{A^3 \exp(At)}{(A\exp(At) - B)^2}, \quad Q_p'(t) = \tau_p Q'(t),$$
$$K'(t) = -\frac{2(n^{(1)} + n^{(2)})(n^{(1)}\tau_1 + n^{(2)}\tau_2)}{[(n^{(1)} + n^{(2)}) + 2(n^{(1)}\tau_1 + n^{(2)}\tau_2)Q(t)]^2} Q'(t). \tag{23}$$

This allows us to check the accuracy of the collision solver without introducing time discretization error.

Table 1 shows the L^∞ norm between the numerical and analytical $\partial f^{(p)}/\partial t$. For different mass ratios, we have considered the cases with $N = \{16, 24, 32, 40, 48, 56\}$ points in each velocity dimension; and $M = 6, 12$ spherical design quadrature points on the full sphere. A good agreement between analytical and numerical solutions is evident from the table. At a fixed N, with increase in mass ratio, the error norm increases. In particular, increase in M does not considerably affect the solution due to the isotropic nature of the distribution function. Note that, in the fast spectral decomposition, since the integral oscillates roughly on $O(N)$, the total number of GaussâĂŞLegendre quadrature points N_ρ in the radial direction should be on order of $O(N)$. As per [4], a more precise estimate is $\approx 0.8N$. However, there is no good rule to select optimal N_ρ. We observe that the error is relatively unaffected upon reducing N_ρ from N to $N/2$. However, we note that $N_\rho = N$ is a safer choice.

From a computational viewpoint, the simulation time is independent of the mass ratio. On increasing the number of discretization points on the sphere M, the computational cost approximately doubles–however, we do observe the effect of loop unrolling for smaller N. Likewise, the computational cost approximately doubles on increasing the number of quadrature points N_ρ. This establishes that the algorithm is linear in both M and N_ρ.

3.3 Spatially in-homogeneous case: Couette flow

The aforementioned methodology allows us to compute the collision operator efficiently. To solve the fully spatial in-homogeneous equation (1), we also need an accurate and efficient spatial and time discretization. Here, we adopt the the Runge-Kutta discontinuous Galerkin (RKDG) approach–widely used for hyperbolic systems–as adapted in [6, 7] for Boltzmann equation. The details of the discretization can be found in [6, 7]. We mention that evaluation of collision operator consumes > 98% of computation time, and hence, in the present work, we focus on the collision operator behavior. More details on spatial-temporal RKDG discretization on GPU can

N	N_ρ	$m_q/m_p = 1$						$m_q/m_p = 4$					
		$M = 6$			$M = 12$			$M = 6$			$M = 12$		
		$time\,(s)$	$\mathcal{E}^{(1)}$	$\mathcal{E}^{(2)}$	$time\,(s)$	$\mathcal{E}^{(1)}$	$\mathcal{E}^{(2)}$	$time\,(s)$	$\mathcal{E}^{(1)}$	$\mathcal{E}^{(2)}$	$time\,(s)$	$\mathcal{E}^{(1)}$	$\mathcal{E}^{(2)}$
16	4	0.00039	3.27e-03	3.27e-03	0.00050	1.77e-03	1.77e-03	0.00039	4.80e-03	1.22e-03	0.00050	3.63e-03	2.47e-04
	8	0.00056	3.73e-03	3.73e-03	0.00085	2.00e-03	2.00e-03	0.00050	4.96e-03	1.33e-03	0.00093	3.62e-03	2.42e-04
	16	0.00084	3.73e-03	3.73e-03	0.00147	2.00e-03	2.00e-03	0.00084	4.96e-03	1.33e-03	0.00148	3.62e-03	2.42e-04
24	6	0.00068	1.37e-04	1.37e-04	0.00114	1.01e-04	1.01e-04	0.00068	1.81e-03	2.06e-02	0.00114	1.79e-03	5.15e-03
	12	0.00117	1.49e-04	1.49e-04	0.00209	9.64e-05	9.64e-05	0.00114	2.12e-03	1.87e-02	0.00210	2.13e-03	6.01e-03
	24	0.00210	1.49e-04	1.49e-04	0.00401	9.64e-05	9.64e-05	0.00210	2.12e-03	1.87e-02	0.00401	2.13e-03	6.01e-03
32	8	0.00159	3.04e-05	3.04e-05	0.00287	2.51e-05	2.51e-05	0.00157	1.54e-04	1.62e-02	0.00286	1.52e-04	1.13e-02
	16	0.00286	3.17e-05	3.17e-05	0.00541	2.45e-05	2.45e-05	0.00286	5.91e-05	1.69e-02	0.00542	5.87e-05	1.03e-02
	32	0.00543	3.17e-05	3.17e-05	0.01057	2.45e-05	2.45e-05	0.00542	5.91e-05	1.69e-02	0.01059	5.87e-05	1.03e-02
40	10	0.00328	1.38e-06	1.38e-06	0.00626	1.26e-06	1.26e-06	0.00326	5.53e-05	4.31e-03	0.00626	5.56e-05	4.35e-03
	20	0.00625	9.35e-07	9.35e-07	0.01226	8.10e-07	8.10e-07	0.00626	5.01e-05	4.29e-03	0.01222	4.97e-05	4.54e-03
	40	0.01227	9.35e-07	9.35e-07	0.02446	8.10e-07	8.10e-07	0.01219	5.01e-05	4.29e-03	0.02431	4.97e-05	4.54e-03
48	12	0.00656	1.04e-07	1.04e-07	0.01289	9.99e-08	9.99e-08	0.00658	8.46e-06	5.76e-04	0.01291	8.45e-06	5.93e-04
	24	0.01290	1.05e-07	1.05e-07	0.02556	9.95e-08	9.95e-08	0.01291	7.17e-06	5.80e-04	0.02561	7.51e-06	6.09e-04
	48	0.02545	1.05e-07	1.05e-07	0.05169	9.95e-08	9.95e-08	0.02550	7.17e-06	5.80e-04	0.05215	7.51e-06	6.09e-04
56	14	0.01204	9.80e-08	9.80e-08	0.02350	9.79e-08	9.79e-08	0.01202	5.22e-06	2.32e-04	0.02352	4.08e-06	1.88e-04
	28	0.02354	9.80e-08	9.80e-08	0.04667	9.79e-08	9.79e-08	0.02353	5.09e-06	2.24e-04	0.04662	3.97e-06	1.87e-04
	56	0.04664	9.80e-08	9.80e-08	0.09303	9.79e-08	9.79e-08	0.04674	5.09e-06	2.24e-04	0.09313	3.97e-06	1.87e-04

Table 1: Efficiency and accuracy L^∞ error $\mathcal{E}^{(p)} = \|\partial_t f^{(p)}_{analytical} - \partial_t f^{(p)}_{numerical}\|$, $p = \{1, 2\}$ for spatially homogeneous Krook-Wu solution at $t = 5.5$ for different mass-ratios. N, N_ρ, and M respectively, denote the number of discretization points in the velocity space, number of Gauss quadrature points in the radial direction, and number of discretization points on full sphere. A fixed velocity domain $[-12,\ 12]^3$ has been used for all the mass-ratios.

be found in [8, 18]. We restrict our discussion and benchmarks to 1-D flow problems for brevity[1].

3.3.1 Verification. For general Boltzmann equation (1), analytical solutions do not exist. Therefore, we compare our results with widely accepted direct simulation Monte Carlo (DSMC) [2] method. We want to emphasize that DSMC is a stochastic method for solution of the N-particle master kinetic equation which converges to the Boltzmann equation in the limit of infinite number of particles [17].

In the current test case, we consider the effect of velocity gradient on the solution. The coordinates are chosen such that the walls are parallel to the y direction and x is the direction perpendicular to the walls. The geometry as well as boundary conditions are shown in Figure 1. Figure 2 illustrates the velocity and temperature along the domain length for both species, wherein we observe an excellent agreement between DGFS and DSMC. The small discrepancies, however, are primarily due to: a) statistical fluctuations inherent to the Monte Carlo methods, b) practical limitations on number of particles used in DSMC simulations. From a computational viewpoint, the present DGFS simulations on a single GPU took 138 seconds to acquire the steady state, in contrast to 26086.45 sec on 24 processors for DSMC simulations as reported in [7], for achieving comparable accuracy.

[1]Discussion and benchmarks for higher 2D/3D spatial dimension shall be presented in the extended version of this manuscript.

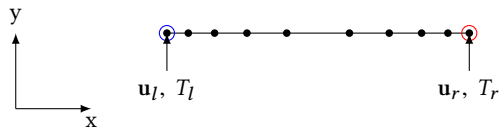


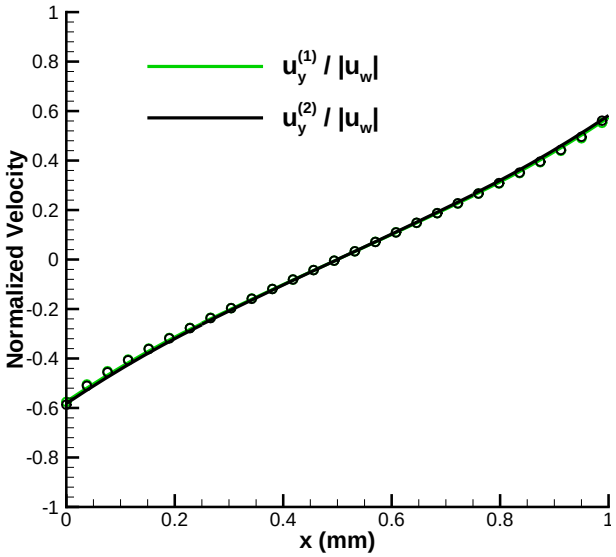
Figure 1: Numerical setup for 1D Couette flow.

3.3.2 Scaling Behavior. The simulations are carried out for different test-cases by varying element-count (N_e), polynomial approximation order ($N_p = K - 1$), and velocity-space sizes (N). The spatial elements are distributed to p processors using the well-known linear domain-decomposition strategy requiring sharing of $O(pN^3)$ floating-point during MPI communication phase. Speed up obtained with multi-GPU solver is presented in Table (3). As evident from the table, the acceleration due to GPU parallelization increases with increase in the size of computational grid. More specifically, the increase in N_e and K have small-effect on overall speedup which suggests that DG-operators (for instance derivative, time-evolution) are rather computationally inexpensive operations. On the other hand, increase in velocity-grid improves the observed speedup. The weak/strong scaling behavior is also evident from the table.

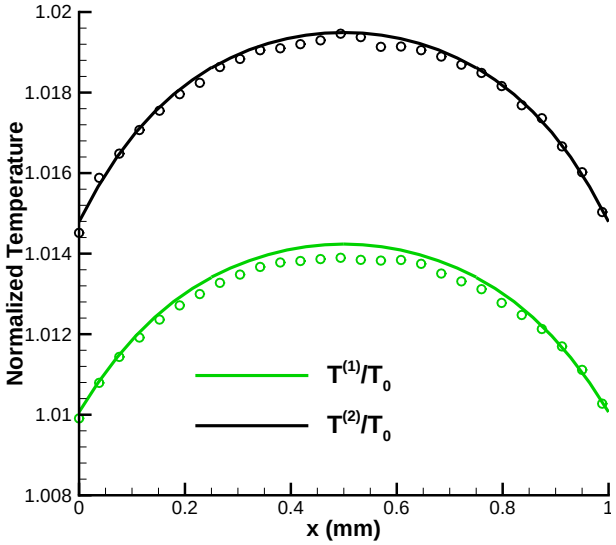
3.3.3 Flat profile. Recall that the fast Fourier spectral collision operator algorithm 2 is split into multiple parts. It is therefore interesting to see what performance level is attained by each part of the operator. Fig (3) presents the percentage of time spent in various parts of Algo. 2 vs. order of DG scheme (K). First, we note

Parameter	Case C-01
Molecular mass: $\{m_1, m_2\}$ $(\times 10^{27}\ kg)$	$\{66.3, 139.1\}$
Non-dim physical space	$[0, 1]$
Non-dim velocity space	$[-7, 7]^3$
$\{N^3, N_\rho, M\}$	$\{32^3, 8, 12\}$
Spatial elements	4
DG order	3
Time stepping	Euler
Viscosity index: $\omega_{\{11,\,12,\,21,\,22\}}$	$\{0.81, 0.805, 0.805, 0.8\}$
Scattering parameter: $\alpha_{\{11,\,12,\,21,\,22\}}$	$\{1.4, 1.36, 1.36, 1.32\}$
Ref. diameter: $d_{\mathrm{ref},pq}$ $(\times 10^{10} m)$	$\{4.11, 4.405, 4.405, 4.7\}$
Ref. temperature: $T_{\mathrm{ref},pq}$ (K)	$\{273\}$
Characteristic mass: m_0 $(\times 10^{27}\ kg)$	66.3
Characteristic length: H_0 (mm)	1
Characteristic velocity: $\mathbf{u}_0$ (m/s)	337.2
Characteristic temperature: T_0 (K)	273
Characteristic number density: n_0 (m^{-3})	1.680×10^{21}
Initial conditions	
Velocity: $\mathbf{u}$ (m/s)	0
Temperature: T (K)	273
Number density: $n^{(1)}$ (m^{-3})	1.680×10^{21}
Number density: $n^{(2)}$ (m^{-3})	8.009×10^{20}
Knudsen number: $(\mathrm{Kn}_{11}, \mathrm{Kn}_{22})$	$(0.793, 0.606)$
Knudsen number: $(\mathrm{Kn}_{12}, \mathrm{Kn}_{21})$	$(0.803, 0.555)$
Left wall (purely diffuse) boundary conditions (subscript l)	
Velocity: $\mathbf{u}_l$ (m/s)	$(0, -50, 0)$
Temperature: T_l (K)	273
Right wall (purely diffuse) boundary conditions (subscript r)	
Velocity: $\mathbf{u}_r$ (m/s)	$(0, +50, 0)$
Temperature: T_r (K)	273

Table 2: Numerical parameters for Couette flow [7]. Based upon our observations from Table 1, we have used $N_\rho = 8$, in contrast to $N_\rho = 32$ used in [7]. This does not affect the recovered bulk properties as illustrated in Fig. 1, however, it speeds up the computation by a factor of 4.



(a) normalized y-component of velocity



(b) normalized temperature

Figure 2: Variation of normalized y-velocity, and temperature along the domain for Couette flow (Case C-01) obtained with DSMC and DGFS using VSS collision kernel for Argon-Krypton mixture. Symbols denote DSMC solutions, and lines denote DGFS solutions.

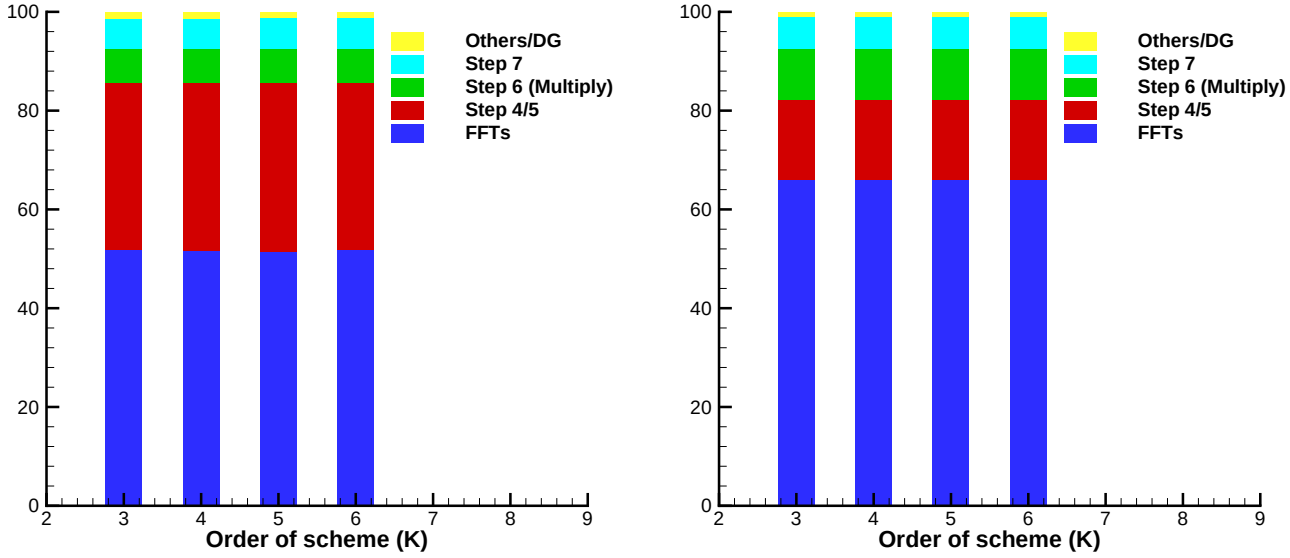
that the DG operators denoted in yellow, requires 1% of the total simulation time. The collision operator, however, consumes nearly > 98% of the total time for both $N^3 = 20^3$ and $N^3 = 32^3$.

4 CONCLUSIONS

We have presented an implementation of the multi-species Discontinuous Galerkin Fast Spectral (DGFS) method for solution of *multi-species* monoatomic full Boltzmann equation on multi-GPU/multi-CPU architectures. The DG-type formulation employed in the present work has advantage of having high-order accuracy at the element-level, and its element-local compact nature (and that of our collision algorithm) enables effective parallelization on massively parallel architectures. For verification and benchmarks, we carry out simulations for spatially homogeneous BKW, and Couette flow problems. Parallel efficiency close to 0.95 is observed on a 36 GPU multi-node/multi-GPU system. An important key observation is that the efficiency can be maintained provided we have enough work on each processor. It is this speedup that now allows researchers to solve problems within a day that would otherwise take months on traditional CPUs. Future work directions include, assessment of the implementation beyond thousand cores. Extending the implementation to general 2D/3D mixed grids coupled with adaptivity in physical and velocity spaces, is an interesting direction as well.

Table 3: Performance of the solver for Couette flow test cases. The phase-space is defined using a convenient triplet notation $\mathbf{N}_e/\mathbf{K}/\mathbf{N}^3$, which corresponds to $\mathbf{N}_e$ elements in physical space, $\mathbf{K}$ order DG (equivalently $N_p = K - 1$ order polynomial for 1-D domain), and $\mathbf{N}^3$ points in velocity space. $n\mathbf{G}$ ($n > 1$) denotes GPU/CUDA/MPI/parallel execution on n GPUs shared equally across ($n/3$) nodes. Work units represent the total simulation time for first 52 timesteps. Efficiency is defined as ratio ($1\mathbf{G}/n\mathbf{G}$)/n, where $1\mathbf{G}$ and $n\mathbf{G}$ are execution-times on one GPU and n GPU respectively. $M = 12$ and $N_\rho = 8$ is used for all cases.

Phase space	Work Units (s)							Efficiency					
	1G	3G	6G	9G	12G	24G	36G	1G/3G	1G/6G	1G/9G	1G/12G	1G/24G	1G/36G
$72/3/20^3$	47.580	16.155	8.339	5.698	4.392	2.423	1.774	0.98	0.95	0.93	0.90	0.82	0.84
$72/3/32^3$	126.601	42.616	21.551	14.563	11.038	5.784	4.030	0.99	0.98	0.97	0.96	0.91	0.98
$72/3/48^3$	391.943	131.081	65.913	44.218	33.513	17.224	11.621	1.00	0.99	0.98	0.97	0.95	1.05
$72/6/20^3$	94.682	31.957	16.197	10.944	8.331	4.392	3.079	0.99	0.97	0.96	0.95	0.90	0.96
$72/6/32^3$	253.016	84.834	42.741	28.697	21.703	11.158	7.693	0.99	0.99	0.98	0.97	0.94	1.03
$72/6/48^3$	782.343	261.601	131.217	87.755	66.009	33.520	22.509	1.00	0.99	0.99	0.99	0.97	1.09
$216/3/20^3$	141.754	47.641	24.033	16.182	12.326	6.356	4.388	0.99	0.98	0.97	0.96	0.93	1.01
$216/3/32^3$	378.956	126.853	63.676	42.636	32.066	16.295	11.041	1.00	0.99	0.99	0.98	0.97	1.07
$216/3/48^3$	1172.907	391.916	196.439	131.153	98.538	49.652	33.471	1.00	1.00	0.99	0.99	0.98	1.10
$216/6/20^3$	283.091	94.737	47.679	31.903	24.060	12.262	8.320	1.00	0.99	0.99	0.98	0.96	1.06
$216/6/32^3$	759.149	253.498	127.004	84.932	63.780	32.212	21.672	1.00	1.00	0.99	0.99	0.98	1.09
$216/6/48^3$	2347.099	783.642	392.470	261.817	196.552	98.680	66.018	1.00	1.00	1.00	1.00	0.99	1.11



(a) $N_e = 72$, $N^3 = 20^3$ **(b)** $N_e = 72$, $N^3 = 32^3$

Figure 3: Percentage of time spent in various parts of Algo. 2 vs. order of DG scheme (K). For both $N^3 = 20^3$ and $N^3 = 32^3$, the collision operator consumes $> 98\%$ of the simulation time.

REFERENCES

[1] Michael Bayer. 2018. Mako: Templates for Python. (2018). https://www.makotemplates.org/

[2] G. A. Bird. 1994. *Molecular Gas Dynamics and the Direct Simulation of Gas Flows.* Clarendon Press, Oxford.

[3] L Bornschein, Katrin-Collaboration, et al. 2005. The KATRIN experiment-a direct measurement of the electron antineutrino mass in the sub-eV region. *Nuclear Physics A* 752 (2005), 14–23.

[4] I. Gamba, J. Haack, C. Hauck, and J. Hu. 2017. A fast spectral method for the Boltzmann collision operator with general collision kernels. *SIAM Journal of Scientific Computing* 39 (2017), B658–B674.

[5] Harold Grad. 1949. On the kinetic theory of rarefied gases. *Communications on pure and applied mathematics* 2, 4 (1949), 331–407.

[6] Shashank Jaiswal, Alina A. Alexeenko, and Jingwei Hu. 2019. A discontinuous Galerkin fast spectral method for the full Boltzmann equation with general collision kernels. *J. Comput. Phys.* 378 (2019), 178–208.

[7] Shashank Jaiswal, Alina A. Alexeenko, and Jingwei Hu. 2019. A discontinuous Galerkin fast spectral method for the multi-species full Boltzmann equation. *Computer Methods in Applied Mechanics and Engineering* (2019). arXiv preprint arXiv:1903.03056.

[8] Andreas Klöckner, Tim Warburton, Jeff Bridge, and Jan S Hesthaven. 2009. Nodal discontinuous Galerkin methods on graphics processors. *J. Comput. Phys.* 228, 21 (2009), 7863–7882.

[9] Katsuhisa Koura and Hiroaki Matsumoto. 1991. Variable soft sphere molecular model for inverse-power-law or Lennard-Jones potential. *Physics of Fluids A: Fluid Dynamics* 3, 10 (1991), 2459–2465.

[10] M. Krook and T. T. Wu. 1977. Exact solution of Boltzmann equations for multicomponent systems. *Physical Review Letters* 38, 18 (1977), 991.

[11] EP Muntz. 1989. Rarefied gas dynamis. *Annual Review of Fluid Mechanics* 21 (1989), 387–417.

[12] L. Pareschi and G. Russo. 2000. Numerical solution of the Boltzmann equation I: spectrally accurate approximation of the collision operator. *SIAM J. Numer. Anal.* 37 (2000), 1217–1245.

[13] Felix Sharipov and Denize Kalempa. 2003. Velocity slip and temperature jump coefficients for gaseous mixtures. I. Viscous slip coefficient. *Physics of Fluids* 15, 6 (2003), 1800–1806.

[14] Felix Sharipov and Denize Kalempa. 2004. Velocity slip and temperature jump coefficients for gaseous mixtures. II. Thermal slip coefficient. *Physics of Fluids* 16, 3 (2004), 759–764.

[15] Felix Sharipov and Denize Kalempa. 2005. Separation phenomena for gaseous mixture flowing through a long tube into vacuum. *Physics of Fluids* 17, 12 (2005), 127102.

[16] Shigeru Takata and François Golse. 2007. Half-space problem of the nonlinear Boltzmann equation for weak evaporation and condensation of a binary mixture of vapors. *European Journal of Mechanics-B/Fluids* 26, 1 (2007), 105–131.

[17] Wolfgang Wagner. 1992. A convergence proof for Bird's direct simulation Monte Carlo method for the Boltzmann equation. *Journal of Statistical Physics* 66, 3 (1992), 1011–1044.

[18] Freddie D Witherden, Antony M Farrington, and Peter E Vincent. 2014. PyFR: An open source framework for solving advection–diffusion type problems on streaming architectures using the flux reconstruction approach. *Computer Physics Communications* 185, 11 (2014), 3028–3040.

[19] R. Womersley. 2016. Symmetric Spherical Designs on the sphere S^2 with good geometric properties, The University of New South Wales. (2016). http://web.maths.unsw.edu.au/~rsw/Sphere/EffSphDes/ss.html