A fast algorithm for the energy space boson Boltzmann collision operator

Jingwei Hu* and Lexing Ying†

Abstract

This paper introduces a fast algorithm for the energy space boson Boltzmann collision operator. Compared to the direct $O(N^3)$ calculation and the previous $O(N^2 \log N)$ method [8], the new algorithm runs in complexity $O(N \log^2 N)$, which is optimal up to a logarithmic factor (N is the number of gird points in energy space). The basic idea is to partition the 3-D summation domain recursively into elementary shapes so that the summation within each shape becomes a special double convolution that can be computed efficiently by the fast Fourier transform. Numerical examples are presented to illustrate the efficiency and accuracy of the proposed algorithm.

Key words. Boson Boltzmann equation, recursive domain decomposition, double convolution, fast Fourier transform.

AMS subject classifications. 35Q20, 44A35, 65T50.

1 Introduction

The quantum Boltzmann equation or Nordheim-Uehling-Uhlenbeck equation [9, 14], describes the non-equilibrium dynamics of quantum gases. These are the low density gases consisting of bosons or fermions which, when cooled to certain temperatures, evolve and interact in ways that reveal the quantum mechanical nature of the particles. Although most of the discussion also applies to the Fermi gas, in this paper we will only focus on the Bose gas since it covers the interesting phenomenon of the Bose-Einstein condensation (BEC).

Under the spatially homogeneous and velocity isotropic assumption, one can derive the following energy space boson Boltzmann equation [11, 12, 5, 2, 8, 13] from its phase space counterpart:

$$\rho(\varepsilon)\frac{\partial f}{\partial t} = Q(f)(\varepsilon), \quad \varepsilon \geq 0, \tag{1.1}$$

where $f(t, \varepsilon)$ is the distribution function that depends on time t and particle energy ε. The function $\rho(\varepsilon)$ is the density of states:

$$\rho(\varepsilon) := \int_{\mathbb{R}^3} \delta\left(\varepsilon - \frac{\mathbf{v}^2}{2}\right) d\mathbf{v} = 4\pi\sqrt{2\varepsilon}. \tag{1.2}$$

*Institute for Computational Engineering and Sciences (ICES), The University of Texas at Austin, 1 University Station, C0200, Austin, TX 78712, USA. hu@ices.utexas.edu.

†Department of Mathematics and ICES, The University of Texas at Austin, 1 University Station, C1200, Austin, TX 78712, USA. lexing@math.utexas.edu.

The quantum collision operator $Q(f)$ models the interaction of bosons:

$$Q(f)(\varepsilon) = \int_0^\infty \int_0^\infty \int_0^\infty w(\varepsilon, \varepsilon_*, \varepsilon', \varepsilon'_*)\delta(\varepsilon + \varepsilon_* - \varepsilon' - \varepsilon'_*) \\ \cdot [f' f'_*(1+f)(1+f_*) - f f_*(1+f')(1+f'_*)]\, d\varepsilon_* d\varepsilon' d\varepsilon'_*, \tag{1.3}$$

where $(\varepsilon, \varepsilon_*)$ and $(\varepsilon', \varepsilon'_*)$ are the particle energies before and after collision. f, f_*, f', and f'_* are shorthand notations for $f(t,\varepsilon)$, $f(t,\varepsilon_*)$, $f(t,\varepsilon')$, and $f(t,\varepsilon'_*)$ respectively. The collision kernel w is a nonnegative function determined by the underlying interaction law. For the simple Maxwellian molecules, w is given by

$$w(\varepsilon, \varepsilon_*, \varepsilon', \varepsilon'_*) = \rho(\min(\varepsilon, \varepsilon_*, \varepsilon', \varepsilon'_*)). \tag{1.4}$$

A brief derivation of (1.1) – (1.4) and a summary of basic properties of the equation can be found in Appendix A.

Compared to the phase space description, the energy space equation is greatly simplified. A lot of theoretical work has been conducted in the past few years for (1.1) and related models, see, for instance, [3, 13, 4, 7, 1] and references therein. For numerical approximations of (1.1), we refer to [11, 12, 5, 2, 8].

The goal of this paper is to design an efficient algorithm for the boson Boltzmann collision operator (1.3). Our starting point is the following truncated version of $Q(f)$ used in [8]:

$$Q^R(f)(\varepsilon) = \int_0^R \int_0^R \int_0^R \rho(\min(\varepsilon, \varepsilon_*, \varepsilon', \varepsilon'_*))\delta(\varepsilon + \varepsilon_* - \varepsilon' - \varepsilon'_*) \\ \cdot [f' f'_*(1+f)(1+f_*) - f f_*(1+f')(1+f'_*)]\, d\varepsilon_* d\varepsilon' d\varepsilon'_*, \quad \varepsilon \in [0, R]. \tag{1.5}$$

How to choose the upper bound R will be made more precise in the numerical results. Here we only mention that in order to capture the physics R is usually not small. We then introduce N uniform discrete points $\varepsilon_0 < \varepsilon_1 < \ldots < \varepsilon_{N-1}$ on $[0, R]$ with mesh size $\Delta\varepsilon = R/N$. Thus a consistent discretization of (1.5) is written as

$$\begin{aligned} Q_i^R &= \Delta\varepsilon^2 \sum_{\substack{m,n,j=0 \\ m+n=i+j}}^{N-1} \rho(\varepsilon_{\min})[f_m f_n (1+f_i)(1+f_j) - f_i f_j (1+f_m)(1+f_n)] \\ &= \Delta\varepsilon^2 (1+f_i) \sum_{\substack{m,n,j=0 \\ m+n=i+j}}^{N-1} \rho(\varepsilon_{\min}) f_m f_n (1+f_j) - \Delta\varepsilon^2 f_i \sum_{\substack{m,n,j=0 \\ m+n=i+j}}^{N-1} \rho(\varepsilon_{\min}) f_j (1+f_m)(1+f_n), \end{aligned} \tag{1.6}$$

for $i = 0, \ldots, N-1$, and

$$\varepsilon_{\min} := \min(\varepsilon_i, \varepsilon_j, \varepsilon_m, \varepsilon_n) = \varepsilon_{\min(m,n,i,j)}. \tag{1.7}$$

This is just a simple numerical quadrature rule that takes into account the approximation of the delta function. Depending on the application, one can either choose integer grid points (first order method), or half-integer grid points (second order method). It is not difficult to verify that the scheme (1.6) preserves the main physical features of the continuous problem: conservation of mass and energy, the entropy inequality, and the Bose-Einstein distribution as steady state. See [8] for more details.

Despite its simple form the efficient evaluation of (1.6) still presents a challenge. Clearly a direct calculation of Q_i (for all i) requires cubic complexity $O(N^3)$, which can be quite expensive for large N. Furthermore, it is well known that a singularity occurs at the origin when the BEC happens, thus a finer grid is necessary to maintain the resolution.

In [8] by exploiting the special form of (1.7), Markowich and Pareschi were able to reduce the above cost to $O(N^2 \log N)$ (all the log in this paper refers to logarithm to base 2). Their approach is based on a 2-D domain decomposition that allows one to use the fast Fourier transform (FFT) to speed up the inner summation – a convolution.

In this work, we propose a faster algorithm for (1.6) that runs in only $O(N \log^2 N)$ steps, which is optimal up to a logarithmic factor. The main idea is to partition the 3-D summation domain recursively into elementary shapes such that the FFT can be applied to both inner and outer summations – a special double convolution.

The rest of the paper is organized as follows: in the next section we describe the fast algorithm in detail and analyze its complexity. Numerical results for computation of the collision operator and the time-evolution equation are shown in Section 3. Finally the concluding remarks are given in Section 4.

2 Fast algorithms for the boson Boltzmann collision operator

We first briefly review the previous $O(N^2 \log N)$ method in [8], since it provides a basis for constructing the new algorithm.

2.1 The previous $O(N^2 \log N)$ algorithm – 2-D domain decomposition

The key observation behind the method [8] is: if one divides the grid $\{0, \ldots, N-1\}^2$ in the index mn – domain into four parts according to a fixed $j \in \{0, \ldots, N-1\}$:

- region (i): $\{(m,n) : \ 0 \le m \le j, \ 0 \le n \le j\}$;
- region (ii): $\{(m,n) : \ j < m \le N-1, \ j < n \le N-1\}$;
- region (iii): $\{(m,n) : \ 0 \le m \le j, \ j < n \le N-1\}$;
- region (iv): $\{(m,n) : \ j < m \le N-1, \ 0 \le n \le j\}$,

(see Figure 1 for an illustration), then $\varepsilon_{\min}$ (1.7) takes a unique value in each region, i.e.,

$$\min(m,n,i,j) = \begin{cases} i & \text{in region } (i); \\ j & \text{in region } (ii); \\ m & \text{in region } (iii); \\ n & \text{in region } (iv). \end{cases}$$

Now the computation of (1.6) can be performed in each region separately. We write

$$Q_i^R = Q_i^{(i)} + Q_i^{(ii)} + Q_i^{(iii)} + Q_i^{(iv)},$$

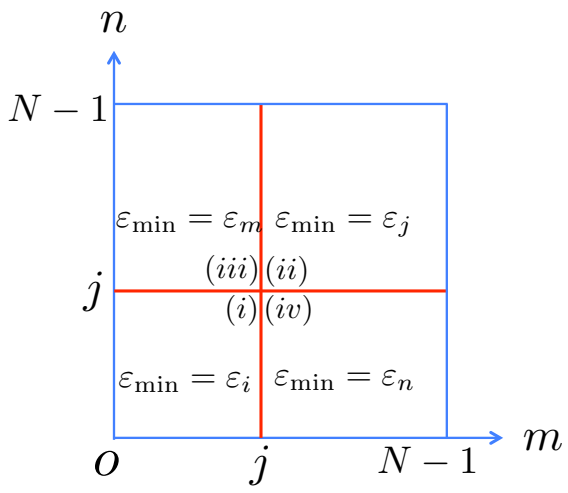

Figure 1: 2-D summation domain decomposition.

where $Q_i^{(i)}$ denotes the summation of m, n in region (i) for each j, and so on so forth. Let us take region (iii) for example, where

$$Q_i^{(iii)} = \Delta\varepsilon^2(1+f_i)\sum_{j=0}^{N-1}\left(\sum_{\substack{0\le m\le j, j<n\le N-1\\ m+n=i+j}}\rho(\varepsilon_m)f_m f_n\right)(1+f_j)$$
$$-\Delta\varepsilon^2 f_i\sum_{j=0}^{N-1}\left(\sum_{\substack{0\le m\le j, j<n\le N-1\\ m+n=i+j}}\rho(\varepsilon_m)(1+f_m)(1+f_n)\right)f_j, \quad \text{for} \quad i=0,\ldots,N-1.$$

For the gain term, if we treat $\rho(\varepsilon_m)f_m$ as a single function, then for each fixed j the inner summation is a convolution of functions ρf and f defined on truncated portions according to j. Similarly, the inner sum of the loss term is a convolution of truncated functions $\rho(1+f)$ and $(1+f)$. Either of them can be computed effectively by the FFT in $O(N\log N)$ operations, resulting a function g with index $i+j$. Since g itself depends on j, the outer summation has to be carried out directly. Therefore, the total cost is $O(N^2\log N + N^2) = O(N^2\log N)$.

2.2 The new $O(N\log^2 N)$ algorithm – 3-D domain decomposition

We observe from (1.6) that the main computation task is of the following general form

$$u_i = \sum_{m+n=i+j}\rho(\varepsilon_{\min})e_m g_n h_j, \quad \text{for} \quad 0\le i,j,m,n\le N-1,$$

where $\{e_m\}$, $\{g_n\}$, and $\{h_j\}$ are sequences indexed by $\{0,\ldots,N-1\}$. It turns out that in order to speed up the calculation, it is more convenient to consider all possible i that satisfy $m+n=i+j$ for some $0\le m,n,j\le N-1$ and then truncate the result to $0\le i\le N-1$. This allows us to consider the *3-D summation domain*

$$D := \{(m,n,j):\ 0\le m,n,j\le N-1\}.$$

As pointed out earlier, the approach of [8] partitions the 2-D summation domain $\{(m,n):\ 0 \le m,n \le N-1\}$ into four regions for each fixed j. We instead partition the whole summation domain D into four regions – two *pyramids* and two *simplexes* (the 3-D counterparts of 2-D regions (i) – (iv), see Figure 2):

- region (I): $\{(m,n,j):\ m \le j,\ n \le j,\ 0 \le m,n,j \le N-1\}$;
- region (II): $\{(m,n,j):\ j < m,\ j < n,\ 0 \le m,n,j \le N-1\}$;
- region (III): $\{(m,n,j):\ m \le j,\ j < n,\ 0 \le m,n,j \le N-1\}$;
- region (IV): $\{(m,n,j):\ j < m,\ n \le j,\ 0 \le m,n,j \le N-1\}$,

and compute the contribution from each region one by one.

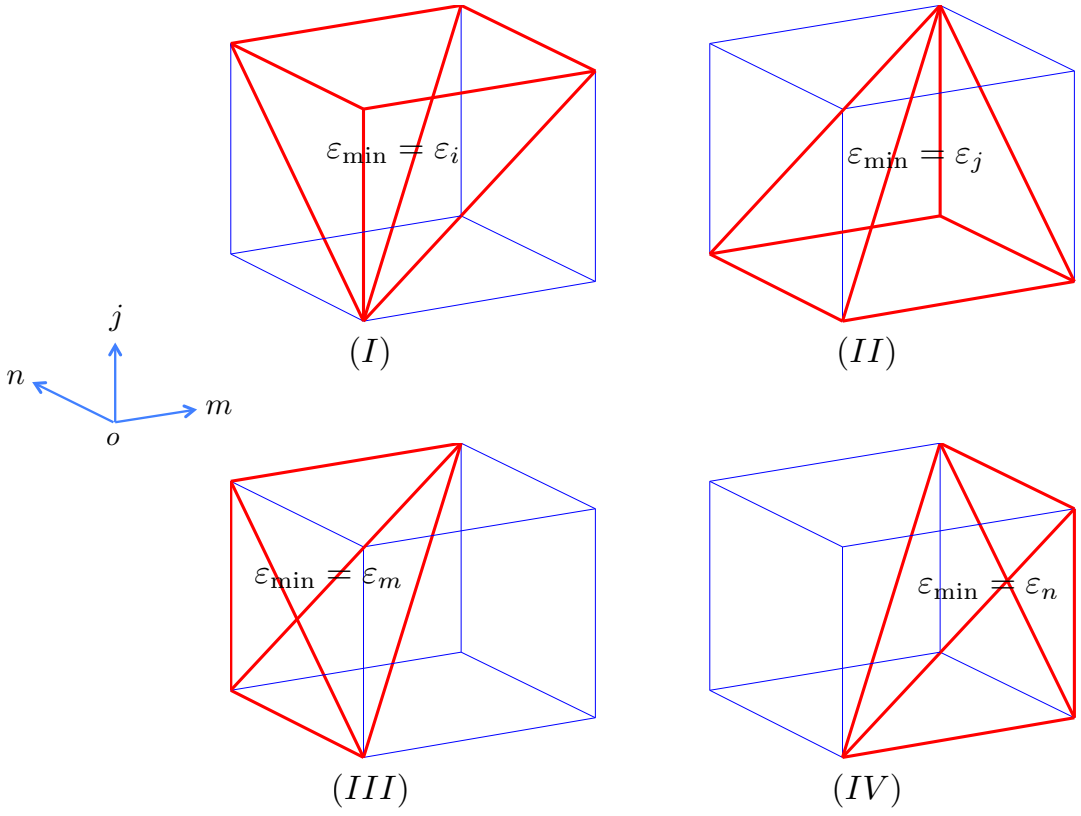

Figure 2: 3-D summation domain decomposition.

Within each 3-D region, $\min(m,n,i,j)$ takes a fixed value

$$\min(m,n,i,j) = \begin{cases} i & \text{in region } (I); \\ j & \text{in region } (II); \\ m & \text{in region } (III); \\ n & \text{in region } (IV), \end{cases}$$

and it is possible to combine the troublesome term $\rho(\varepsilon_{\min})$ with one of the sequences $\{e_m\}$, $\{g_n\}$, $\{h_j\}$ or treat it as an extra multiplication. Hence, we only need to consider the following simple problem of the form

$$u_i = \sum_{m+n=i+j} e_m g_n h_j$$

with the summation taken over a single region and $\{e_m\}$, $\{g_n\}$, and $\{h_j\}$ are again defined on $\{0, \ldots, N-1\}$ (but with slightly different content).

Before discussing the algorithm in detail, we note that the next three summations will be encountered often later on. Given three sequences $\{e_m\}$, $\{g_n\}$, and $\{h_j\}$ defined on three independent consecutive integer intervals $\mathcal{M}$, $\mathcal{N}$, and $\mathcal{J}$ respectively, let

$$p_i = \sum_{\substack{m\in\mathcal{M}, n\in\mathcal{N} \\ i=m+n}} e_m g_n, \quad i \in \mathcal{M}+\mathcal{N}, \tag{2.1}$$

$$q_i = \sum_{\substack{m\in\mathcal{M}, n\in\mathcal{N} \\ i=m-n}} e_m g_n, \quad i \in \mathcal{M}-\mathcal{N}, \tag{2.2}$$

$$r_i = \sum_{\substack{m\in\mathcal{M}, n\in\mathcal{N}, j\in\mathcal{J} \\ i=m+n-j}} e_m g_n h_j, \quad i \in \mathcal{M}+\mathcal{N}-\mathcal{J}, \tag{2.3}$$

where $\mathcal{M}+\mathcal{N} = \{m+n:\ m\in\mathcal{M},\ n\in\mathcal{N}\}$, $\mathcal{M}-\mathcal{N} = \{m-n:\ m\in\mathcal{M},\ n\in\mathcal{N}\}$, and $\mathcal{M}+\mathcal{N}-\mathcal{J} = \{m+n-j:\ m\in\mathcal{M},\ n\in\mathcal{N},\ j\in\mathcal{J}\}$. We claim that the sequences $\{p_i\}$, $\{q_i\}$, and $\{r_i\}$ can all be evaluated by fast algorithms. In fact,

- (2.1) is nothing but a simple convolution of $\{e_m\}$ and $\{g_n\}$, which can be calculated by the FFT.
- To compute (2.2), we first reverse the ordering of $\{g_n\}$, i.e., introduce a new sequence $\{\tilde{g}_n = g_{-n}\}$ with $n \in -\mathcal{N}$, and rewrite q_i as

  $$q_i = \sum_{\substack{m\in\mathcal{M}, n\in-\mathcal{N} \\ i=m+n}} e_m g_{-n} = \sum_{\substack{m\in\mathcal{M}, n\in-\mathcal{N} \\ i=m+n}} e_m \tilde{g}_n, \quad i \in \mathcal{M}-\mathcal{N}.$$

  This again falls into the form (2.1).
- For (2.3), we introduce the sequence $\{\tilde{h}_j = h_{-j}\}$ with $j \in -\mathcal{J}$, then

  $$r_i = \sum_{\substack{m\in\mathcal{M}, n\in\mathcal{N}, j\in-\mathcal{J} \\ i=m+n+j}} e_m g_n \tilde{h}_j, \quad i \in \mathcal{M}+\mathcal{N}-\mathcal{J}.$$

  This is a double convolution and can also be sped up by the FFT.

Clearly for any of the above problems the computational cost is bounded by

$$\max(|\mathcal{M}|, |\mathcal{N}|, |\mathcal{J}|)\log(\max(|\mathcal{M}|, |\mathcal{N}|, |\mathcal{J}|)),$$

where $|\cdot|$ denotes the cardinality of a set. We are ready to describe the final algorithm.

Computation of region (I) For region (I), the relevant sum is of form

$$u_i^I = \sum_{\substack{m\le j, n\le j, 0\le m,n,j\le N-1 \\ m+n=i+j}} e_m g_n h_j.$$

In order to compute it efficiently, we partition this pyramid into five parts as shown in Figure 3.

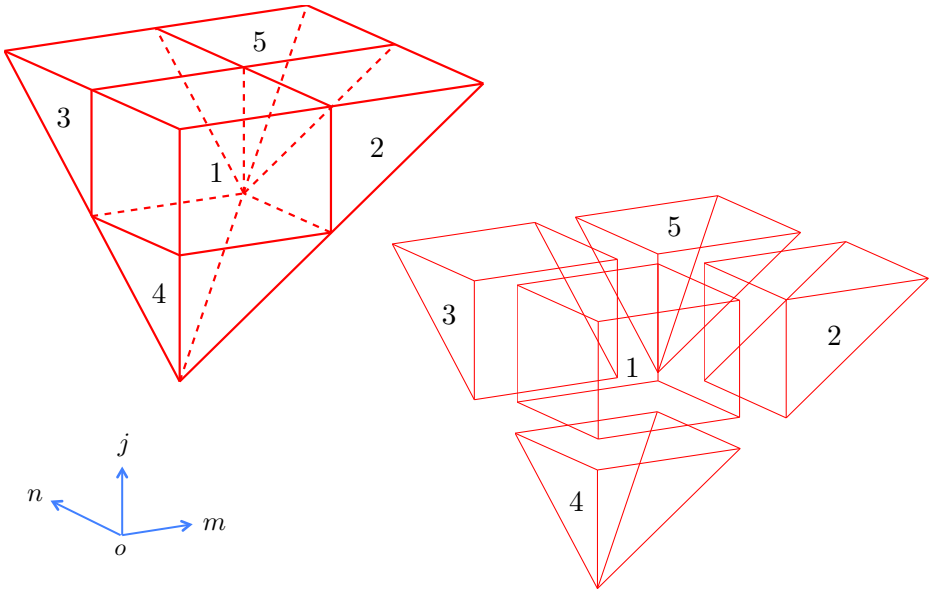

Figure 3: Decomposition of pyramid (I).

- Part 1 is a cube:

$$\left\{(m,n,j):\ 0\le m,n\le \frac{N}{2}-1,\ \frac{N}{2}\le j\le N-1\right\}.$$

- Part 2 is a wedge:

$$\left\{(m,n,j):\ 0\le n\le \frac{N}{2}-1,\ \frac{N}{2}\le m\le j\le N-1\right\}.$$

- Part 3 is another wedge:

$$\left\{(m,n,j):\ 0\le m\le \frac{N}{2}-1,\ \frac{N}{2}\le n\le j\le N-1\right\}.$$

- Part 4 is a pyramid:

$$\left\{(m,n,j):\ m\le j,\ n\le j,\ 0\le m,n,j\le \frac{N}{2}-1\right\}.$$

- Part 5 is another pyramid:

$$\left\{(m,n,j):\ m\le j,\ n\le j,\ \frac{N}{2}\le m,n,j\le N-1\right\}.$$

We can now write $u_i^I = u_i^{I,1} + u_i^{I,2} + u_i^{I,3} + u_i^{I,4} + u_i^{I,5}$, where each $u_i^{I,k}$ stands for the summation related to the kth part.

- For part 1 (cube), the relevant sum is

$$u_i^{I,1} = \sum_{\substack{0 \le m,n \le \frac{N}{2}-1, \frac{N}{2} \le j \le N-1 \\ m+n=i+j}} e_m g_n h_j.$$

  This double convolution can be computed as for (2.3) and the cost is $O\left(\frac{N}{2}\log\frac{N}{2}\right)$.

- For part 2 (wedge), the relevant sum is

$$u_i^{I,2} = \sum_{n=0}^{\frac{N}{2}-1} \underbrace{\left(\sum_{\substack{\frac{N}{2} \le m \le j \le N-1 \\ m-j=i-n}} e_m h_j \right)}_{t_{i-n}} g_n.$$

  The inner summation corresponds to the triangular side of the wedge. At first sight, this summation is difficult to compute due to the existence of the constraint $m \le j$. However, notice that $m \le j$ is equivalent to $i - n \le 0$ since $m - j = i - n$. Therefore, all one needs is to compute the inner sum as for (2.2) and set the resulting vector t_{i-n} to zero at indices greater than 0. The outer summation is computed as for (2.1). Thus the total cost for this part is also $O\left(\frac{N}{2}\log\frac{N}{2}\right)$.

- For part 3 (wedge), the relevant sum is

$$u_i^{I,3} = \sum_{m=0}^{\frac{N}{2}-1} \left(\sum_{\substack{\frac{N}{2} \le n \le j \le N-1 \\ n-j=i-m}} g_n h_j \right) e_m.$$

  This is computed exactly the same as above for part 2.

- Parts 4 and 5 (pyramids) have the same shape as region (I), but only half its size. It is thus natural to perform the computation of these two parts using recursion.

Let us denote the computational cost for region (I) by $T(N)$, where N stands for the size of the summation domain in each dimension. We have

$$\begin{aligned} T(N) &= \underbrace{O\left(\frac{N}{2}\log\frac{N}{2}\right)}_{\text{one cube}} + \underbrace{2O\left(\frac{N}{2}\log\frac{N}{2}\right)}_{\text{two wedges}} + \underbrace{2T\left(\frac{N}{2}\right)}_{\text{two pyramids}} = O\left(N\log N\right) + 2T\left(\frac{N}{2}\right) \\ &= \ldots = \underbrace{O\left(N\log N + N\log\frac{N}{2} + N\log\frac{N}{4} + \ldots\right)}_{\log N \text{ terms}} + NT(1) \\ &= O(N\log^2 N) + O(N) = O(N\log^2 N). \end{aligned}$$

In the actual implementation, we terminate the recursion whenever the size of the pyramid is smaller than a certain threshold (e.g. 64 or 128), since then the quadratic-complexity algorithm [8] becomes competitive.

Computation of region (II) As the shape of region (II) is similar to the one of region (I), it can be decomposed as a disjoint union of five parts: one cube, two wedges, and two self-similar pyramids of half the original size (see Figure 4). Therefore, the approach discussed for region (I) works with minor modifications. As a result, the computational cost of region (II) is also $O(N \log^2 N)$.

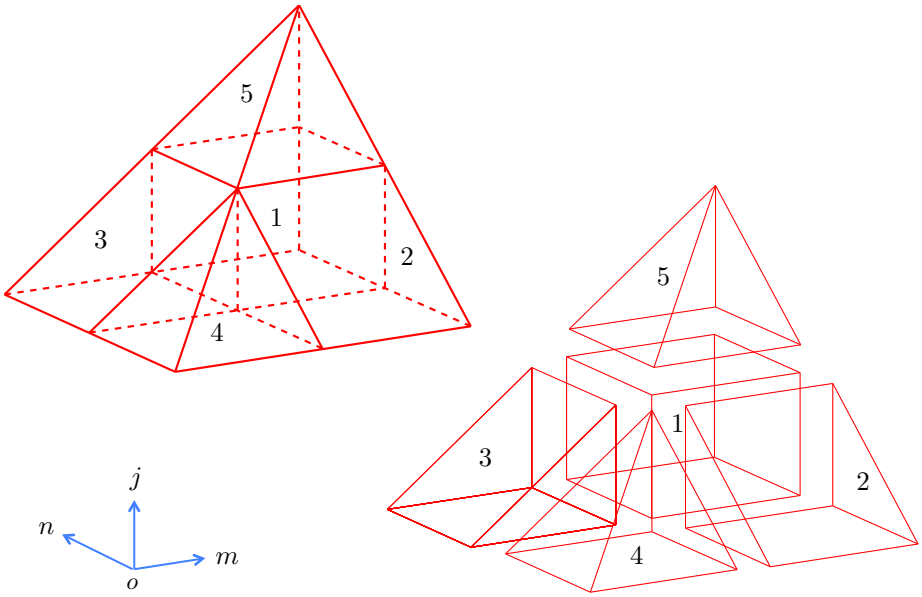

Figure 4: Decomposition of pyramid (II).

Computation of region (III) For region (III), the relevant sum is of form

$$u_i^{III} = \sum_{\substack{m \le j, j < n, 0 \le m,n,j \le N-1 \\ m+n=i+j}} e_m g_n h_j.$$

We partition this simplex into four parts as shown in Figure 5.

- Part 1 is a wedge:

$$\left\{(m,n,j):\ \frac{N}{2} \le n \le N-1,\ 0 \le m \le j \le \frac{N}{2}-1\right\}.$$

- Part 2 is another wedge:

$$\left\{(m,n,j):\ 0 \le m \le \frac{N}{2}-1,\ \frac{N}{2} \le j < n \le N-1\right\}.$$

- Part 3 is a simplex:

$$\left\{(m,n,j):\ m \le j,\ j < n,\ 0 \le m,n,j \le \frac{N}{2}-1\right\}.$$

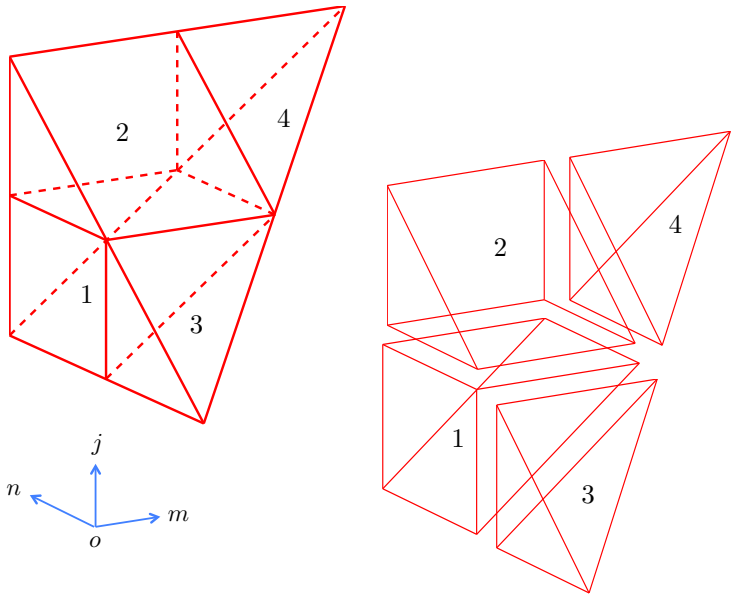

Figure 5: Decomposition of simplex (III).

- Part 4 is another simplex:

$$\left\{(m,n,j):\ m\le j,\ j<n,\ \frac{N}{2}\le m,n,j\le N-1\right\}.$$

We can write $u_i^{III}=u_i^{III,1}+u_i^{III,2}+u_i^{III,3}+u_i^{III,4}$ accordingly, so that each $u_i^{III,k}$ stands for the summation associated with the kth part.

- In part 1 (wedge), the relevant sum is

$$u_i^{III,1}=\sum_{n=\frac{N}{2}}^{N-1}\left(\sum_{\substack{0\le m\le j\le \frac{N}{2}-1\\ m-j=i-n}} e_m h_j\right) g_n.$$

  Similarly to part 2 in region (I), we compute the inner sum as for (2.2) and set the value of the result to zero at indices greater than 0. The outer sum is computed as for (2.1). Thus the total cost for this part is $O\left(\frac{N}{2}\log\frac{N}{2}\right)$.

- In part 2 (wedge), the relevant sum is

$$u_i^{III,2}=\sum_{m=0}^{\frac{N}{2}-1}\left(\sum_{\substack{\frac{N}{2}\le j<n\le N-1\\ n-j=i-m}} g_n h_j\right) e_m.$$

  This one is computed almost the same as above for part 1, with the only exception that we set the result of the inner sum to zero at indices less than or equal to 0 (since now $n-j=i-m>0$).

- Parts 3 and 4 (simplexes) have the same shape as region (III) but only half its size, so the computation associated with them can be done using recursion again.

Assume the computational cost for region (III) is $T(N)$, where N is the size of the summation domain in each dimension. Then

$$T(N) = \underbrace{2O\left(\frac{N}{2}\log\frac{N}{2}\right)}_{\text{two wedges}} + \underbrace{2T\left(\frac{N}{2}\right)}_{\text{two simplexes}},$$

which again yields an algorithm of $O(N\log^2 N)$.

Computation of region (IV) As the shape of region (IV) is similar to that of region (III), it can be decomposed as a disjoint union of four parts: two wedges, and two self-similar simplexes of half the original size (see Figure 6). Therefore, the approach discussed for region (III) works with minor modifications. As a result, the cost of region (IV) is also $O(N\log^2 N)$.

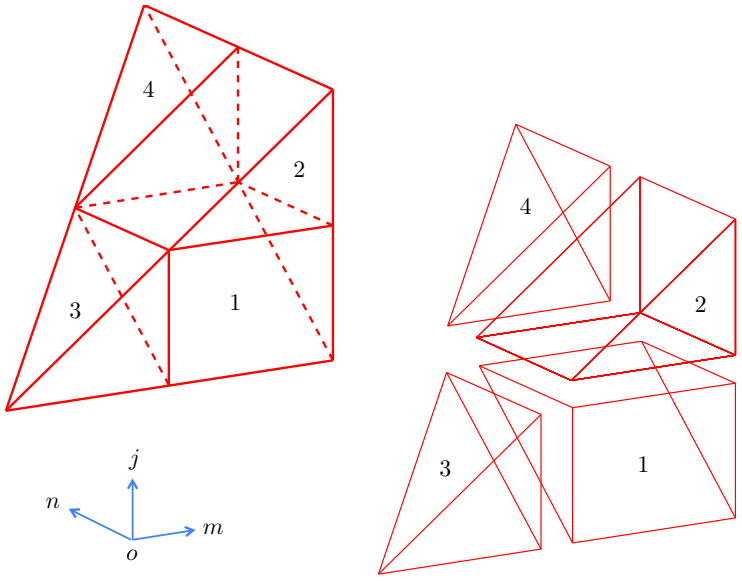

Figure 6: Decomposition of simplex (IV).

3 Numerical results

In this section we provide several numerical examples to demonstrate the efficiency and accuracy of the new algorithm. We first test its performance on the collision operator, and then use it to solve the time-evolution equation. In all the examples, the grid points are chosen as $\varepsilon_0 = \Delta\varepsilon/2$, $\varepsilon_1 = 3\Delta\varepsilon/2$, ..., $\varepsilon_{N-1} = R - \Delta\varepsilon/2$ (second-order quadrature rule).

3.1 Computing the collision operator

We first test the proposed algorithm on a single application of the distribution

$$f(\varepsilon) = 3\exp\left(-(\varepsilon - 10)^2\right), \quad \varepsilon \in [0, R], \quad R = 30.$$

Here $R = 30$ is chosen such that $R \geq 2R_0$, where $\mathrm{supp}(f) \approx [0, R_0]$. This guarantees the disregarded value is significantly small by conservation of energy (see Figure 7).

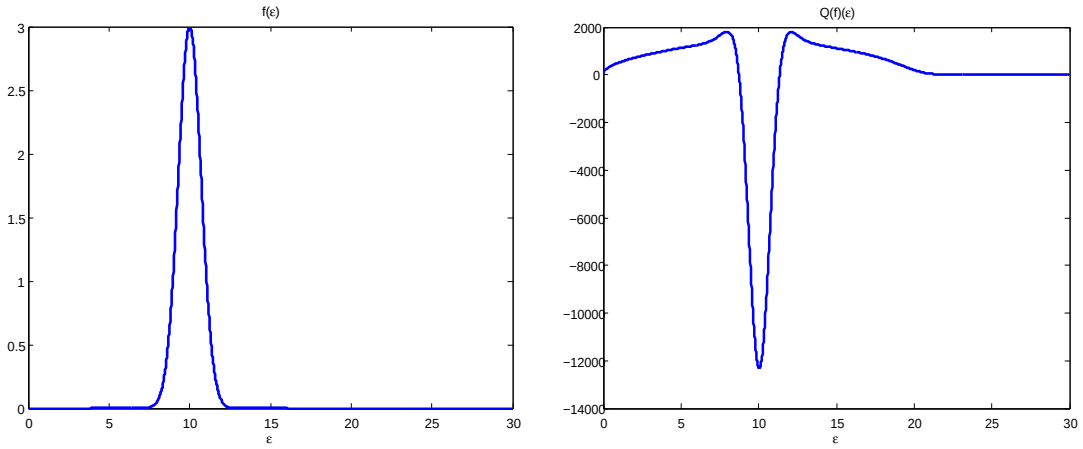

Figure 7: Left: the distribution function $f(\varepsilon)$. Right: the numerical $Q(f)(\varepsilon)$ computed by the new algorithm. $N = 2048$.

The results for different N are reported in Table 1. For comparison, the method in [8] is referred to as *quadratic algorithm*. Column 5 confirms the linear complexity of the new algorithm.

N	quadratic algorithm T_q	$\frac{T_q(N)}{T_q(N/2)}$	new algorithm T_n	$\frac{T_n(N)}{T_n(N/2)}$	$\|Q_n - Q_q\|_2/\|Q_q\|_2$
128	0.386s	–	0.278s	–	3.2620e-16
256	1.450s	3.76	0.510s	1.83	8.2386e-16
512	5.579s	3.85	0.926s	1.82	1.2436e-15
1024	22.334s	4.00	1.816s	1.96	1.9140e-15
2048	88.980s	3.98	3.696s	2.04	2.0821e-15

Table 1: The average running time and the relative error of the quadratic algorithm in [8] and our new algorithm.

The speedup factor over the quadratic algorithm is about 24 for $N = 2048$.

3.2 Solving the boson Boltzmann equation

We now solve the equation (1.1) explicitly by a second-order Runge-Kutta method. The new fast algorithm is applied to the collision operator. We will only consider the Bose gas in non-degenerate regime ($z < 1$). Modeling the degenerate Bose gas ($z > 1$) is a very complicated issue due the singularity of the distribution function (A.8) at the origin and is beyond the scope of this paper.

Suppose the initial condition is given by

$$f_0(\varepsilon) = e^{-(\varepsilon-10)^2/10}, \quad \varepsilon \in [0, R], \quad R = 120.$$

The corresponding equilibrium takes the form (A.6) with $z \approx 0.6336$, $\beta \approx 0.1236$.

Remark 3.1 *The reason to choose such a large R is because the final equilibrium $\mathcal{M}_{z,\beta}(\varepsilon)$ has a larger compact support. Moreover, the functions $\rho(\varepsilon)\mathcal{M}_{z,\beta}(\varepsilon)$ and $\rho(\varepsilon)\varepsilon\mathcal{M}_{z,\beta}(\varepsilon)$ (the integrands of mass and energy (A.3), (A.4)) spread even wider (see Figure 8). So in order to capture the real physics, R has to be large enough to include all mass and energy.*

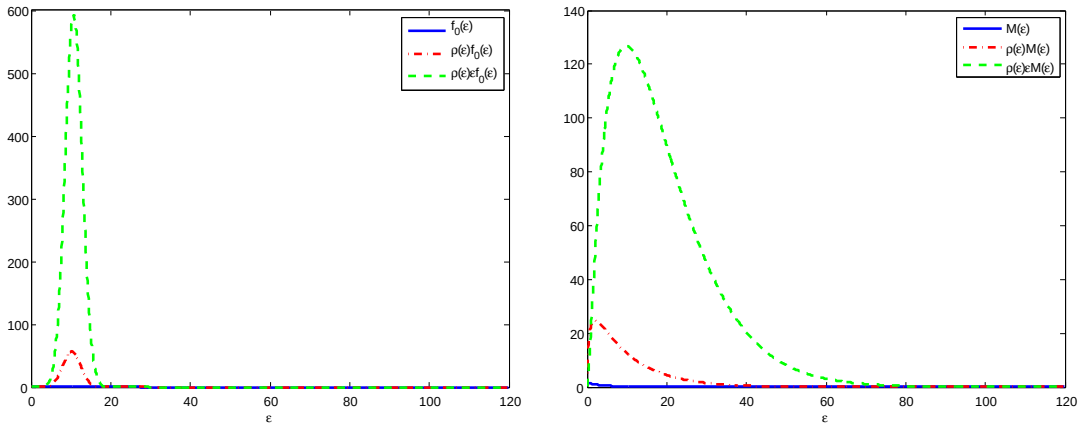

Figure 8: Left: the initial distribution $f_0(\varepsilon)$, $\rho(\varepsilon)f_0(\varepsilon)$, and $\rho(\varepsilon)\varepsilon f_0(\varepsilon)$. Right: the exact equilibrium $\mathcal{M}_{z,\beta}(\varepsilon)$, $\rho(\varepsilon)\mathcal{M}_{z,\beta}(\varepsilon)$, and $\rho(\varepsilon)\varepsilon\mathcal{M}_{z,\beta}(\varepsilon)$.

Figure 9 shows the distribution function $f(t, \varepsilon)$ at different times together with the exact equilibrium $\mathcal{M}_{z,\beta}(\varepsilon)$. The conservation of mass and energy, and the entropy growth can be observed from Figure 10. Here the number of grid points is $N = 2048$. The time step size is $\Delta t = 0.005$.

At certain time, we expect the numerical solution $f(t, \varepsilon)$ converges to the exact equilibrium $\mathcal{M}_{z,\beta}(\varepsilon)$ within an acceptable error. This can be seen from Table 2.

N	$\|f - \mathcal{M}_{z,\beta}\|_2 / \|\mathcal{M}_{z,\beta}\|_2$	extrapolated value of $f(0)$
128	0.0121	1.6894
256	0.0047	1.7168
512	0.0017	1.7252
1024	0.0006	1.7279
2048	0.0002	1.7289

Table 2: The relative error of f and $\mathcal{M}_{z,\beta}$ at fixed time $t = 0.5$ ($\Delta t = 0.005$) and the cubic spline extrapolated value of $f(0)$ (Ref: the exact value $\mathcal{M}_{z,\beta}(0) = 1.7294$).

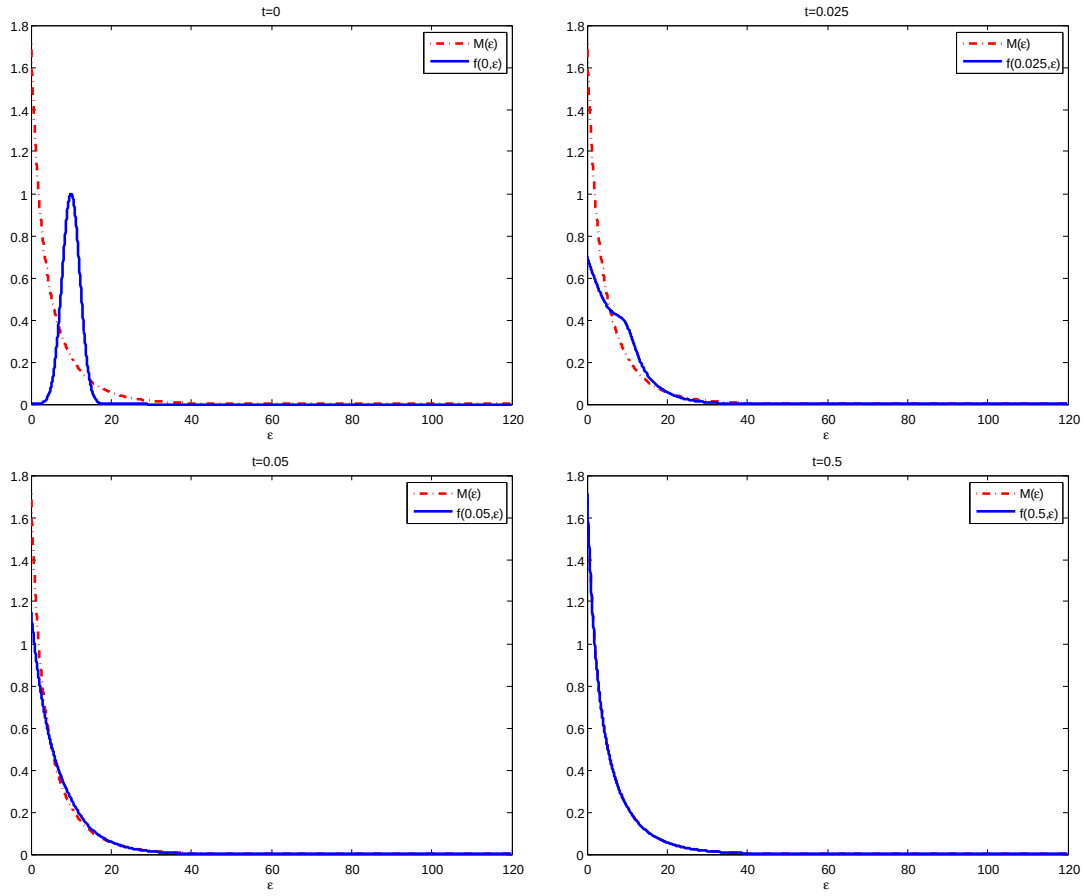

Figure 9: The numerical solution f at $t = 0$, 0.025, 0.05, and 0.5, and the exact equilibrium $\mathcal{M}_{z,\beta}$. $N = 2048$. $\Delta t = 0.005$.

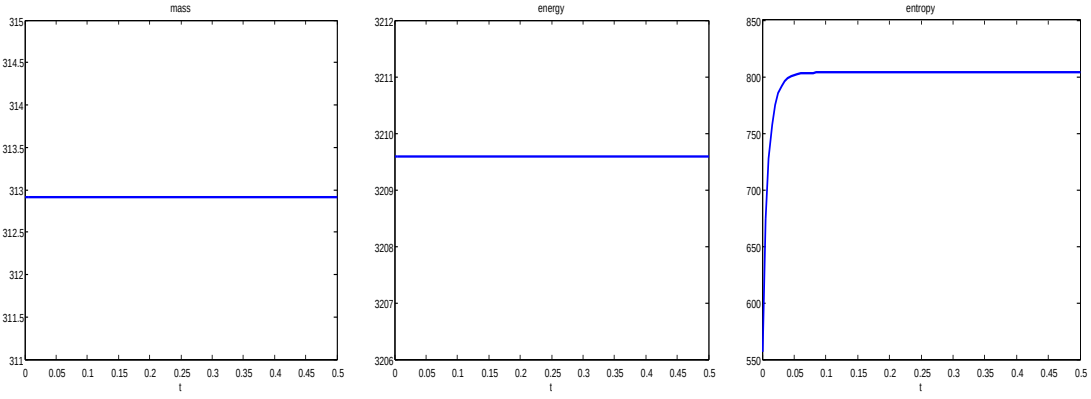

Figure 10: The time evolution of mass, energy, and entropy. $N = 2048$. $\Delta t = 0.005$.

4 Conclusions and future work

A fast linear (up to a logarithmic factor) algorithm is constructed for the energy space boson Boltzmann collision operator. The idea is to decompose the 3-D summation domain to

elementary shapes: cubes, wedges, and self-similar pyramids and simplexes such that the collision kernel is a constant within each region. The summations in the cubes and wedges are double convolutions that can be evaluated efficiently with FFTs, while the self-similar parts are treated recursively, yielding an algorithm of $O(N \log^2 N)$. The numerical results further confirm the linear complexity of the method and demonstrate its robustness in solving the time-evolution equation.

So far we have not touched the degenerate Bose gas yet. It seems that a logarithmic grid is more suitable to describe the condensation at the origin. The design of fast algorithms in this framework will be investigated in the future.

Acknowledgments. J.H. is supported by an ICES Postdoctoral Fellowship. L.Y. is partially supported by NSF under CAREER award DMS-0846501. We thank Prof. Irene Gamba and the kinetic theory group at UT-Austin for helpful discussion.

Appendix A The energy space boson Boltzmann equation

In this Appendix, we shall give a formal derivation of the energy space boson Boltzmann equation (1.1) from its original form in the phase space, and summarize its basic properties. Part of the arguments can be found in one place or another such as [12, 3, 8, 13, 6, 10].

We will start from the spatially homogeneous quantum Boltzmann equation in [9, 14] (a similar derivation carries through for the spatially inhomogeneous case and when there is an external potential). Let $F(t, \mathbf{v})$ be the phase space distribution function of time t and particle velocity $\mathbf{v}$, then the equation reads:

$$\frac{\partial F}{\partial t} = \tilde{Q}(F)(\mathbf{v}), \quad \mathbf{v} \in \mathbb{R}^3. \tag{A.1}$$

The quantum collision operator $\tilde{Q}(F)$ for a Bose gas is given by (assume a unit mass for all particles)

$$\begin{aligned}\tilde{Q}(F)(\mathbf{v}) = \int_{\mathbb{R}^3}\int_{\mathbb{R}^3}\int_{\mathbb{R}^3} W(\mathbf{v}, \mathbf{v}_*, \mathbf{v}', \mathbf{v}'_*)\delta\left(\mathbf{v} + \mathbf{v}_* - \mathbf{v}' - \mathbf{v}'_*\right)\delta\left(\frac{\mathbf{v}^2}{2} + \frac{\mathbf{v}_*^2}{2} - \frac{\mathbf{v}'^2}{2} - \frac{\mathbf{v}'^2_*}{2}\right) \\ \cdot \left[F'F'_*(1+F)(1+F_*) - FF_*(1+F')(1+F'_*)\right] d\mathbf{v}_* d\mathbf{v}' d\mathbf{v}'_*,\end{aligned} \tag{A.2}$$

where $(\mathbf{v}, \mathbf{v}_*)$ and $(\mathbf{v}', \mathbf{v}'_*)$ are the velocity pairs before and after collision. As usual, F, F_*, F', and F'_* stand for $F(t, \mathbf{v})$, $F(t, \mathbf{v}_*)$, $F(t, \mathbf{v}')$, and $F(t, \mathbf{v}'_*)$. The collision kernel W depends on the specific interaction law. Here we consider the simple case of $W = 1$ which corresponds to Maxwellian molecules.

We now make the assumption that $F(t, \mathbf{v})$ is isotropic in $\mathbf{v}$, i.e., one can define an energy space distribution $f(t, \varepsilon)$ such that $f(t, \varepsilon) = F(t, \mathbf{v})$ with $\varepsilon = \mathbf{v}^2/2$. Then by the change of variables, the total mass of particles (per unit volume) is

$$M := \int_{\mathbb{R}^3} F(t, \mathbf{v})\, d\mathbf{v} = \int_0^\infty 4\pi\sqrt{2\varepsilon}\, f(t, \varepsilon)\, d\varepsilon, \tag{A.3}$$

and the total energy is

$$E := \int_{\mathbb{R}^3} \frac{\mathbf{v}^2}{2} F(t, \mathbf{v})\, d\mathbf{v} = \int_0^\infty 4\pi\sqrt{2\varepsilon}\, \varepsilon f(t, \varepsilon)\, d\varepsilon. \tag{A.4}$$

Therefore, it is convenient to introduce the function $\rho(\varepsilon)$ as in (1.2).

Similarly the collision operator (A.2) can be transformed into

$$\tilde{Q}(F) = \frac{4\pi^2}{\rho(\varepsilon)} \int_0^\infty \int_0^\infty \int_0^\infty \rho(\min(\varepsilon, \varepsilon_*, \varepsilon', \varepsilon'_*))\delta\left(\varepsilon + \varepsilon_* - \varepsilon' - \varepsilon'_*\right) \cdot \left[f' f'_* (1+f)(1+f_*) - f f_* (1+f')(1+f'_*)\right] d\varepsilon_* d\varepsilon' d\varepsilon'_*, \tag{A.5}$$

where the equality

$$\int_{\mathbb{S}^2} \int_{\mathbb{S}^2} \int_{\mathbb{S}^2} \delta\left(r\sigma + r_*\sigma_* - r'\sigma' - r'_*\sigma'_*\right)\, d\sigma_* d\sigma' d\sigma'_* = \frac{4\pi^2}{r r_* r' r'_*} \min(r, r_*, r', r'_*),$$

($\mathbf{v} = r\sigma$, $r = |\mathbf{v}|$ and σ is the surface element) and the definition of $\rho(\varepsilon)$ have been used.

Plugging (A.5) back into (A.1), we get the energy space boson Boltzmann equation (1.1) – (1.4) considered at the beginning of the paper (differ by a constant).

A.1 Properties

The quantum collision operator (1.3) has 1 and ε as collision invariants:

$$\int_0^\infty Q(f)\, d\varepsilon = \int_0^\infty \varepsilon Q(f)\, d\varepsilon = 0,$$

so the total mass and energy are conserved:

$$M = \int_0^\infty \rho(\varepsilon)\, f(t,\varepsilon)\, d\varepsilon \equiv \int_0^\infty \rho(\varepsilon)\, f_0(\varepsilon)\, d\varepsilon,$$
$$E = \int_0^\infty \rho(\varepsilon)\, \varepsilon f(t,\varepsilon)\, d\varepsilon \equiv \int_0^\infty \rho(\varepsilon)\, \varepsilon f_0(\varepsilon)\, d\varepsilon,$$

where $f_0(\varepsilon)$ is the initial condition.

$Q(f)$ also satisfies the Boltzmann's H-theorem:

$$\frac{d}{dt} S(\varepsilon) = \int_0^\infty Q(f)[\ln(1+f) - \ln f]\, d\varepsilon \geq 0,$$

where

$$S(\varepsilon) := \int_0^\infty \rho(\varepsilon)[(1+f)\ln(1+f) - f\ln f]\, d\varepsilon$$

is the entropy.

The entropy is always increasing, and reaches its maximum if and only if f attains the equilibrium (the Bose-Einstein distribution):

$$\mathcal{M}_{(z,\beta)}(\varepsilon) = \frac{1}{z^{-1}e^{\beta\varepsilon} - 1}, \tag{A.6}$$

where z is the fugacity, β is the inverse temperature ($\beta = 1/T$; $z = e^{\mu/T}$, μ is the chemical potential). Given $\mathcal{M}_{(z,\beta)}$, the corresponding mass and energy can be expressed as

$$\begin{cases} M = \left(\dfrac{2\pi}{\beta}\right)^{\frac{3}{2}} G_{\frac{3}{2}}(z), \quad z \leq 1, \\ \\ E = \dfrac{3}{2\beta}\left(\dfrac{2\pi}{\beta}\right)^{\frac{3}{2}} G_{\frac{5}{2}}(z), \quad z \leq 1, \end{cases}$$

where $G_\nu(z)$ is the Bose-Einstein function of order ν:

$$G_\nu(z) = \frac{1}{\Gamma(\nu)} \int_0^\infty \frac{x^{\nu-1}}{z^{-1}e^x - 1} dx, \quad 0 < z < 1,\ \nu > 0;\ z = 1,\ \nu > 1. \tag{A.7}$$

For small z, the integrand in (A.7) can be expanded in powers of z,

$$G_\nu(z) = \sum_{n=1}^{\infty} \frac{z^n}{n^\nu} = z + \frac{z^2}{2^\nu} + \frac{z^3}{3^\nu} + \dots.$$

Thus the Bose gas behaves like a classical gas when $z \ll 1$. On the other hand, it becomes degenerate as $z \to 1$.

The famous BEC happens when $z > 1$. The equilibrium state $\mathcal{M}_{(z,\beta)}$ is then composed of two parts (in the sense of maximizing entropy):

$$\mathcal{M}_{(z,\beta)}(\varepsilon) = \frac{1}{e^{\beta\varepsilon} - 1} + \frac{\ln z}{\rho(\varepsilon)} \delta(\varepsilon). \tag{A.8}$$

The physical meaning of z is not fugacity anymore; it is an indicator of the condensate mass. Correspondingly, M and E are given by

$$\begin{cases} M = \left(\dfrac{2\pi}{\beta}\right)^{\frac{3}{2}} G_{\frac{3}{2}}(1) + \ln z, \quad z > 1, \\ \\ E = \dfrac{3}{2\beta} \left(\dfrac{2\pi}{\beta}\right)^{\frac{3}{2}} G_{\frac{5}{2}}(1), \quad z > 1. \end{cases}$$

Note that $G_\nu(1)$ is just the Riemann-Zeta function $\zeta(\nu)$ convergent for $\nu > 1$. In particular, $G_{3/2}(1) \approx 2.6124$, $G_{5/2}(1) \approx 1.3415$.

References

[1] L. Arkeryd and A. Nouri. Bose condensates in interaction with excitations: a kinetic model. *Commun. Math. Phys.*, 310:765–788, 2012.

[2] C. Connaughton and Y. Pomeau. Kinetic theory and Bose-Einstein condensation. *C. R. Pysique*, 5:91–106, 2004.

[3] M. Escobedo, S. Mischler, and M. A. Valle. Homogeneous Boltzmann equation in quantum relativistic kinetic theory. *J. Differ. Equ*, Monograph 4, 2003.

[4] M. Escobedo, F. Pezzotti, and M. Valle. Analytical approach to relaxation dynamics of condensed Bose gases. *Ann. Phys.*, 326:808–827, 2011.

[5] R. Lacaze, P. Lallemand, Y. Pomeau, and S. Rica. Dynamical formation of a Bose-Einstein condensate. *Physica D*, 152–153:779–786, 2001.

[6] L. D. Landau and E. M. Lifshitz. *Statistical Physics*, volume 5 of *Course of Theoretical Physics*. Butterworth-Heinemann, third edition, 1980.

[7] X. Lu and X. Zhang. On the Boltzmann equation for 2D Bose-Einstein particles. *J. Stat. Phys.*, 143:990–1019, 2011.

[8] P. Markowich and L. Pareschi. Fast, conservative and entropic numerical methods for the Boson Boltzmann equation. *Numerische Math.*, 99:509–532, 2005.

[9] L. W. Nordheim. On the kinetic method in the new statistics and its application in the electron theory of conductivity. *Proc. R. Soc. London, Ser. A*, 119:689–698, 1928.

[10] R. K. Pathria. *Statistical Mechanics*. Butterworth-Heinemann, second edition, 1996.

[11] D. V. Semikoz and I. I. Tkachev. Kinetics of Bose condensation. *Phys. Rev. Lett.*, 74:3093–3097, 1995.

[12] D. V. Semikoz and I. I. Tkachev. Condensation of bosons in the kinetic regime. *Phys. Rev. D*, 55:489–502, 1997.

[13] H. Spohn. Kinetics of the Bose-Einstein condensation. *Physica D*, 239:627–634, 2010.

[14] E. A. Uehling and G. E. Uhlenbeck. Transport phenomena in Einstein-Bose and Fermi-Dirac gases. I. *Phys. Rev.*, 43:552–561, 1933.