Adjoint DSMC for Nonlinear Spatially-Homogeneous Boltzmann Equation With a General Collision Model

Yunan Yang[a,*], Denis Silantyev[b], Russel Caflisch[c]

[a]*Institute for Theoretical Studies, ETH Zurich, 8092 Zurich, Switzerland.*
[b]*Department of Mathematics, University of Colorado, Colorado Springs, CO 80918.*
[c]*Courant Institute of Mathematical Sciences, New York University, New York, NY 10012.*

Abstract

We derive an adjoint method for the Direct Simulation Monte Carlo (DSMC) method for the spatially homogeneous Boltzmann equation with a general collision law. This generalizes our previous results in [Caflisch, R., Silantyev, D. and Yang, Y., 2021. Journal of Computational Physics, 439, p.110404], which was restricted to the case of Maxwell molecules, for which the collision rate is constant. The main difficulty in generalizing the previous results is that a rejection sampling step is required in the DSMC algorithm in order to handle the variable collision rate. We find a new term corresponding to the so-called score function in the adjoint equation and a new adjoint Jacobian matrix capturing the dependence of the collision parameter on the velocities. The new formula works for a much more general class of collision models.

1. Introduction

Kinetic equations have gained great popularity in the past three decades as modeling tools beyond their classical application regime of statistical physics [10, 20, 8]. Evolution-type equations for the statistical function of the car, human, or animal positions in a framework similar to the kinetic theory of gases have been widely used to model the dynamics of traffic, human crowds, and swarms [1]. In particular, the Boltzmann equation, mostly known to model rarefied gas, has been studied as the energy-transport model for semiconductors [22, 6], financial Brownian motion [14], wealth distribution [21], and sea ice dynamics [11].

Due to the rise in computational power and the capability to collect an enormous amount of data, data-driven modeling has become a practical mainstream approach. An essential component of data-driven modeling is optimization, where the physical or nonphysical model parameters are regarded as the optimizer of an objective function that depends on the solution of the forward kinetic model [2, 12]. There is usually no explicit formula for the dependence between the model parameter and the solution to the Boltzmann equation [3]. Moreover, the large dimensionality of the unknown parameter makes it impractical to perform a global search to solve the corresponding optimization problem. Due to these two major

*Corresponding author

Email addresses: yunan.yang@eth-its.ethz.ch (Yunan Yang), dsilanty@uccs.edu (Denis Silantyev), caflisch@courant.nyu.edu (Russel Caflisch)

challenges, one often turns to gradient-based optimization algorithms to perform large-scale model parameter calibration. A computationally efficient method to compute the gradient is thus essential, as iterative optimization algorithms such as gradient descent may require hundreds of gradient evaluations in a large-scale optimization task.

In an earlier work [9], we investigated the spatially homogeneous Boltzmann equation constrained optimization problems and developed Monte Carlo algorithms to compute the gradient based on the two common approaches in the context of PDE-constrained optimization: optimize-then-discretize (OTD) and discretize-then-optimize (DTO). Since the Boltzmann equation is an integro-differential equation, so is the corresponding adjoint equation. Obtaining the gradient based on the OTD approach requires numerical solutions to both the forward and adjoint integro-differential equations. Although the forward Boltzmann equation can be solved efficiently by the so-called Direct Simulation Monte Carlo (DSMC) method [7, 18, 5, 4, 19], solving the adjoint equation using Monte Carlo-type methods requires interpolation in the three-dimensional velocity space at each time step [9]. As a more efficient alternative, the DTO approach computes the gradient by deriving the adjoint of the Monte Carlo discretization for the forward model. This approach gave rise to the so-called adjoint DSMC method proposed in [9]. One highlight is that the adjoint DSMC method costs even less than the forward DSMC method since it does not require further sampling by using the same sampled velocity pairs and collision parameters from the forward DSMC. The idea of adjoint DSMC was recently extended to a spatially inhomogeneous case in combination with the density method for topology optimization problems [13].

The framework in [9] is based on the spatially homogeneous Boltzmann equation for the Maxwell molecules. That is, the collision kernel in the Boltzmann equation is a constant. In this work, we generalize the adjoint DSMC method to apply to more general collision models for the Boltzmann equation that have both velocity and angle dependence. The forward DSMC method typically uses the rejection sampling method (also known as the acceptance-rejection method) for collision kernels with velocity dependence to draw velocity pairs and the collision scattering angle from a general collision kernel. As the discrete adjoint of the forward DSMC algorithm, the adjoint DSMC will also be modified to reflect the rejection sampling process. Our new generalized adjoint DSMC method is based on the Monte Carlo gradient reparameterization coupled with the rejection sampling algorithms [17, 16]. For collision models with angle dependence, the Jacobian matrix of the post-collision velocities with respect to the pre-collision velocities is also different from the case where the scattering angle is uniformly distributed. Combining the generalization in these two directions, the resulting back-propagation rule for the adjoint particles is a slight modification compared to the case in [9] for the Maxwell molecules.

We can obtain the generalized adjoint DSMC algorithm through two different derivation approaches with the same final result. One is based on the direct approach by differentiating the objective function directly, and the discrete adjoint variable is interpreted as an influence function, which is the derivative with respect to the conditional expectation. The second one is a Lagrangian approach where we impose the forward DSMC collision rules through Lagrangian multipliers (i.e., the discrete adjoint variables) and thus treat velocity particles at all times to be independent. The Karush–Kuhn–Tucker (KKT) conditions lead to the same gradient formulation and adjoint equations as the direct approach. The value of the two derivations is that the direct approach shows the adjoint variable is the influence function,

while the Lagrangian approach should be easier to generalize to other situations.

The derivation of the adjoint equations for collision kernels that depend on the relative velocity requires a modification of the DTO approach, and the resulting method is a mixture of DTO and OTD. As discussed in Section 3.1.4 and Section 5, the velocity derivative (an optimization step) is applied to the expectation over the rejection sampling step, and then sampling (i.e., choice of the relevant random variables) for the rejection sampling (a discretization step) is performed afterward. For all other steps of the method, the DTO approach is followed since discretization is performed before differentiation.

The rest of the paper is organized as follows. We first present some essential background in Section 2 where we briefly review the spatially homogeneous Boltzmann equation with a general collision kernel and the DSMC method with provable convergence for solving such Boltzmann equations. In Section 3, two complementary approaches to deriving the adjoint DSMC method are presented: a direct optimization approach, in Section 3.1 and an augmented Lagrangian approach in Section 3.2. The direct approach is easier to derive and to understand; while the Lagrangian approach is more general to apply to other applications. We then present the adjoint Jacobian matrix calculation for a common form of collision kernel in Section 3.3. It is followed by a discussion in Section 3.4 on two special cases of the collision kernel where the adjoint DSMC algorithm could be further simplified. We show two numerical examples in Section 4 for the hard sphere collision models, one without dependence on the scattering angle and one with the dependence. We demonstrate that the adjoint DSMC method computes the gradient accurately up to the standard Monte Carlo error. Conclusion follows in Section 5.

2. Background

In this section, we briefly review the spatially homogeneous Boltzmann equation and the DSMC method for a general collision kernel.

2.1. The Spatially Homogeneous Boltzmann Equation with a General Collision Kernel

Consider the spatially homogeneous Boltzmann equation

$$\frac{\partial f}{\partial t} = Q(f, f). \tag{1}$$

The nonlinear collision operator $Q(f, f)$, which describes the binary collisions among particles, is defined as

$$Q(f,f) = \int_{\mathbb{R}^3}\int_{\mathcal{S}^2} q(v - v_1, \sigma)(f(v_1')f(v') - f(v_1)f(v))d\sigma dv_1,$$

in which (v', v_1') represent the post-collisional velocities associated with the pre-collisional velocities (v, v_1), $q \geq 0$, and the σ integral is over the surface of the unit sphere $\mathcal{S}^2$.

By conserving the momentum $v + v_1$ and the energy $v^2 + v_1^2$, we have

$$\begin{aligned} v' &= 1/2(v + v_1) + 1/2|v - v_1|\sigma, \\ v_1' &= 1/2(v + v_1) - 1/2|v - v_1|\sigma, \end{aligned} \tag{2}$$

where $\sigma \in \mathcal{S}^2$ is a collision parameter. We will hereafter use the shorthand notation

$$\begin{aligned}(f, f_1, f', f_1') &= (f(v), f(v_1), f(v'), f(v_1')),\\ (\hat{f}, \hat{f}_1, \hat{f}', \hat{f}_1') &= \rho^{-1}(f(v), f(v_1), f(v'), f(v_1'))\end{aligned}$$

for the values of f in the Boltzmann collision operator Q, as well as the normalized densities $\hat{f}$ with $\rho = \int f(v)dv$ and $\int \hat{f}(v)dv = 1$.

Physical symmetries imply that

$$q(v - v_1, \sigma) = \tilde{q}(|v - v_1|, \theta) \tag{3}$$

where $\cos\theta = \sigma \cdot \alpha$ with $\alpha = \frac{v-v_1}{|v-v_1|}$ and $\sigma = \frac{v'-v_1'}{|v'-v_1'|} = \frac{v'-v_1'}{|v-v_1|}$. In [9], we rewrote (2), in terms of operators, as

$$\begin{pmatrix} v' \\ v_1' \end{pmatrix} = A(\sigma, \alpha) \begin{pmatrix} v \\ v_1 \end{pmatrix}, \quad \begin{pmatrix} v \\ v_1 \end{pmatrix} = B(\sigma, \alpha) \begin{pmatrix} v' \\ v_1' \end{pmatrix}, \tag{4}$$

where

$$A(\sigma, \alpha) = \frac{1}{2} \begin{pmatrix} I + \sigma\alpha^T & I - \sigma\alpha^T \\ I - \sigma\alpha^T & I + \sigma\alpha^T \end{pmatrix}, \; B(\sigma, \alpha) = \frac{1}{2} \begin{pmatrix} I + \alpha\sigma^T & I - \alpha\sigma^T \\ I - \alpha\sigma^T & I + \alpha\sigma^T \end{pmatrix}, \tag{5}$$

where I is the identity matrix in $\mathbb{R}^3$. Note that $B = A^T = A^{-1}$, showing the involutive nature of the collision. We also define the matrix $C \in \mathbb{R}^6$ as the derivative of the collision outcome (v', v_1') with respect to the incoming velocities (v, v_1), i.e.,

$$C(\sigma, \alpha) = \frac{\partial(v', v_1')}{\partial(v, v_1)}. \tag{6}$$

Our main assumption on the collision kernel q is that there is a positive constant Σ such that

$$q(v - v_1, \sigma) \leq \Sigma \tag{7}$$

for all v, v_1, and σ. In practice, for example, for the numerical solution by the forward DSMC method, this is not a restriction because one can take Σ to be the largest value of q for the discrete set of velocity values.

Although the formulation and analysis presented here are valid for the general collision model (3), the numerical results will be restricted to collision models of the form

$$q(v - v_1, \sigma) = \tilde{q}(|v - v_1|, \theta) = C_\kappa(\theta)|v - v_1|^\beta. \tag{8}$$

We refer to [Sec. 1.4][23] for more modeling intuition regarding this type of collision models. The general model (8) accommodates both the variable hard sphere (VHS) model, when $C_\kappa(\theta)$ is constant, and the variable soft sphere (VSS) model [15] with angular dependence. As described in Section 3.4, additional efficiency is achieved by using the separable dependence of q on $|v - v_1|$ and θ; see Algorithm 3. We remark that σ should be considered as a function of θ, v, v_1 as $\theta = \arccos(\sigma \cdot \alpha)$ based on (8).

With the assumption (7), the Boltzmann equation (1) can be further written as

$$\begin{aligned}\frac{\partial f}{\partial t} &= \int_{\mathbb{R}^3}\int_{\mathcal{S}^2}(f'f_1' - ff_1)\,q(v-v_1,\sigma)d\sigma dv_1 \\ &= \iint f'f_1' q d\sigma dv_1 - \iint ff_1 q d\sigma dv_1 \\ &= \iint f'f_1' q d\sigma dv_1 + \iint ff_1(\Sigma - q)d\sigma dv_1 - \iint \Sigma f f_1 d\sigma dv_1 \\ &= \iint f'f_1' q d\sigma dv_1 + f\iint f_1(\Sigma - q)d\sigma dv_1 - \mu f, \end{aligned} \tag{9}$$

where $\rho = \int f_1 dv_1$ is the density, $A = \int_0^{2\pi}\int_0^{\pi}\sin(\theta)d\theta d\varphi = 4\pi$ is the surface area of the unit sphere, and

$$\mu = A\Sigma\rho. \tag{10}$$

If we multiply both sides of (9) by an arbitrary test function $\phi(v)$, then divide by ρ and finally integrate over v, we obtain

$$\begin{aligned}\frac{\partial(\int \phi\hat{f}dv)}{\partial t} &= \rho\iiint \phi\hat{f}'\hat{f}_1' q d\sigma dv_1 dv + \rho\iiint \phi\hat{f}\hat{f}_1(\Sigma - q)d\sigma dv_1 dv - \mu\int\phi\hat{f}dv \\ &= \rho\iiint \phi'\hat{f}\hat{f}_1 q d\sigma dv_1 dv + \rho\iiint \phi\hat{f}\hat{f}_1(\Sigma - q)d\sigma dv_1 dv - \mu\int\phi\hat{f}dv,\end{aligned}$$

using $dvdv_1 = dv'dv_1'$ and then interchanging notation (v, v_1) and (v', v_1') in the first integral. If we apply the explicit Euler time integration from t_k to $t_{k+1} = t_k + \Delta t$, we obtain

$$\int \phi\ \hat{f}(v,t_{k+1})dv = (1-\Delta t\mu)\int \phi\ \hat{f}(v,t_k)dv + \tag{11}$$

$$\iiint \phi'\ \Delta t\mu\Sigma^{-1}q\ \hat{f}(v,t_k)\hat{f}_1(v_1,t_k)A^{-1}d\sigma dv_1 dv + \tag{12}$$

$$\iiint \phi\ \Delta t\mu\Sigma^{-1}(\Sigma - q)\ \hat{f}(v,t_k)\hat{f}_1(v_1,t_k)A^{-1}d\sigma dv_1 dv. \tag{13}$$

Note that for a fixed k, $\hat{f}(v,t_k) = f(v,t_k)/\rho$ is the probability density of v, and that $\hat{f}(v,t_k)\hat{f}_1(v_1,t_k)A^{-1}$ is the probability density over the three variables (v, v_1, σ). The three terms (11), (12) and (13) on the right-hand side of this equation represent the sampling of the collision process, using rejection sampling, as described in the DSMC algorithm in Section 2.2.

2.2. The DSMC Method

In the DSMC method [7, 18, 4], we consider a set of N velocities evolving in discrete time due to collisions whose distribution can be described by the distribution function f in (1). We divide time interval $[0, T]$ into M number of sub-intervals of size $\Delta t = T/M$. At the k-th time interval, the particle velocities are represented as

$$\mathcal{V}^k = \{v_1, \ldots, v_N\}(t_k),$$

Algorithm 1 DSMC Algorithm Using Rejection Sampling for a General Collision Kernel q

Compute the initial velocity particles based on the initial condition, $\mathcal{V}^0 = \{v_1^0, \ldots, v_N^0\}$.
for $k = 0$ to $M - 1$ **do**
    Given $\mathcal{V}^k$, choose $N_c = \lceil \Delta t \mu N/2 \rceil$ velocity pairs (i_ℓ, i_{ℓ_1}) uniformly without replacement. The remaining $N - 2N_c$ particles do not have a virtual (or real) collision and set $v_i^{k+1} = v_i^k$.
    for $\ell = 1$ to N_c **do**
        Sample σ_ℓ^k uniformly over $\mathcal{S}^2$.
        Compute $\theta_\ell^k = \arccos(\sigma_\ell^k \cdot \alpha_\ell^k)$ and $q_\ell^k = q(|v_{i_\ell}^k - v_{i_{\ell_1}}^k|, \theta_\ell^k)$.
        Draw a random number ξ_ℓ^k from the uniform distribution $\mathcal{U}([0,1])$.
        if $\xi_\ell^k \le q_\ell^k / \Sigma$ **then**
            Perform real collision between $v_{i_\ell}^k$ and $v_{i_{\ell_1}}^k$ following (2) and obtain $(v_{i_\ell}^{k\prime}, v_{i_{\ell_1}}^{k\ \prime})$.
            Set $(v_{i_\ell}^{k+1}, v_{i_{\ell_1}}^{k+1}) = (v_{i_\ell}^{k\prime}, v_{i_{\ell_1}}^{k\ \prime})$.
        else
            The virtual collision is not a real collision. Set $(v_{i_\ell}^{k+1}, v_{i_{\ell_1}}^{k+1}) = (v_{i_\ell}^k, v_{i_{\ell_1}}^k)$.
        end if
    end for
end for

and we denote the i-th velocity particle in $\mathcal{V}^k$ as v_i^k. The distribution function $f(v,t)$ is then discretized by the empirical distribution

$$f(v, t_k) \approx \frac{\rho}{N} \sum_{i=1}^{N} \delta(v - v_i^k), \quad k = 0, \ldots, M. \tag{14}$$

We define the total number of virtual collision pairs $N_c = \lceil \Delta t \mu N/2 \rceil$. Note that the number of particles having a virtual collision is $2N_c \approx \Delta t \mu N$. Thus, the probability of having a virtual collision is $\Delta t \mu$, and the probability of not having a virtual collision is $1 - \Delta t \mu$. For each velocity $v_i^k \in \mathcal{V}^k$ in this algorithm, there are three possible outcomes, whose probabilities are denoted by h_j where $j = 1, 2, 3$:

Outcome	**Probability**
1. No virtual collision	$h_1 = 1 - \Delta t \mu$
2. A real collision	$h_2 = \Delta t \mu \ q_i^k / \Sigma$
3. A virtual, but not a real, collision	$h_3 = \Delta t \mu (1 - q_i^k / \Sigma)$

Here, $q_i^k = q(v_i^k - v_{i_1}^k, \sigma_i^k)$ where $v_{i_1}^k$ represents the virtual collision partner of v_i^k and σ_i^k is the sampled collision parameter for this pair. Note that the total probability of no real collision is $h_1 + h_3 = 1 - \Delta t \mu q_i^k / \Sigma$. Since the h_j's depend on the collision kernel q, we may also view them each as function of $v - v_1$ and σ. In later sections, we will use the fact that

$$\begin{aligned}
\partial_{v_i^k}(\log h_1) &= 0, \\
\partial_{v_i^k}(\log h_2) &= (q_i^k)^{-1} \partial_{v_i^k} q_i^k, \\
\partial_{v_i^k}(\log h_3) &= -(\Sigma - q_i^k)^{-1} \partial_{v_i^k} q_i^k,
\end{aligned}$$

since ρ, Δt and Σ are constants. Also note that $\partial_{v_{i_1}^k}(\log h_j) = -\partial_{v_i^k}(\log h_j)$.

We can decide whether a particle participates in a *virtual* collision or not through uniform sampling. However, in order to determine whether a selected virtual collision pair participates in the *actual* collision or not, we need to use the rejection sampling since the probability q_i^k is pair-dependent. We further remark that if a virtual velocity pair is rejected for a real collision (Outcome #2), it is automatically accepted for a virtual but not real collision (Outcome #3). With all the notations defined above, we present the DSMC algorithm using the rejection sampling in Algorithm 1. The algorithm applies to any general collision kernel satisfying (7).

Note that the DSMC algorithm is a single sample of the dynamics for N particles following equations (11)-(13). It is not the same as N samples of a single particle because of the nonlinearity of these equations. For consistency, the derivation of the adjoint equations also will employ a single sample of the N-particle dynamics.

3. The Adjoint DSMC Method for a General Kernel

Consider an optimization problem for the spatially homogeneous Boltzmann equation (1). The initial condition is

$$f(v,0) = f_0(v;m), \tag{15}$$

where f_0 is the prescribed initial data depending on a parameter m. The goal is to find m which optimizes the objective function at time $t = T$,

$$J(m) = \int_{\mathbb{R}^3} \phi(v) f(v,T) dv, \tag{16}$$

where $f(v,T)$ is the solution to (1) given the initial condition (15), and thus $f(v,T)$ depends on m through the initial condition.

The adjoint DSMC method proposed in [9] is an efficient particle-based method to compute the gradient of the objective function (16) based on the forward DSMC scheme (Section 2.2). In this work, we generalize the setup in [9] and extend the algorithm to a general collision kernel. This section presents two different ways to obtain the general adjoint DSMC algorithm: a direct approach and a Lagrangian multiplier approach.

3.1. A Direct Approach

In this subsection, we present a direct approach by directly differentiating an equivalent form of the objective function (16) by rewriting it as the expectation over N particles at time $t = T$ (see eq. (19) below).

3.1.1. Expectations for DSMC with N Particles

Here we define the expectations for each step of the forward DSMC algorithm. For simplicity, we assume that the number of particles N is even so that the number of particle pairs is $N/2$. If N is odd, there is no real difference since the single unpaired particle does not get involved in any collisions. The velocities change at a discrete time in the DSMC Algorithm 1. This involves the following two steps at time t_k where $k = 0, \ldots, M-1$.

The first step is to randomly (and uniformly) select collision pairs, v_i^k and $v_{i_1}^k$, and also to randomly select collision parameters. The expectation over this step will be denoted as E_p^k (with "p" signifying collision "parameters" and collision "pairs").

The second step is to perform collisions at the correct rate, i.e., the given collision kernel, $q(v - v_i, \sigma)$, using the rejection sampling. This is performed by choosing outcome j with probability $h_{ij}^k = h_j(v_i^k - v_{i_1}^k, \sigma_i^k)$ for $j = 1, 2, 3$ for this collision pair $(v_i^k, v_{i_1}^k)$; see Section 2.2 for the definition of $\{h_j\}$. The expectation for this step will be denoted as E_r^k (with "r" signifying collision "rejection sampling"). Given all velocity particles at t_k, the total expectation over the step from t_k to t_{k+1} is

$$E^k = E_p^k E_r^k.$$

The introduction of E_r^k here is motivated by the related work on calculating Monte Carlo gradients where the samples are drawn using the rejection sampling [16, 17].

Moreover, the expectation operator E_r^k can each be factored into a series of expectations,

$$E_r^k = \Pi_{i=1}^{N/2} E_{ri}^k, \tag{17}$$

in which the expectations E_{ri}^k is the expectation with respect to the rejection sampling process applied to a single pair $(v_i^k, v_{i_1}^k)$. These are "pair-wise" products, since the subscript i denotes the i-th velocity pair. Note that in these products, the order of expectations does not matter since their application to a pair $(v_i^k, v_{i_1}^k)$ does not affect other pairs. We will later use the factorization of E_r^k in (17). We do not factorize E_p^k since the velocity pairs are drawn without replacement, so it is not pair-wise i.i.d. process.

We define E^k, E_p^k, E_r^k, E_{ri}^k as expectations over the independent, uniform random variables (rather than the physical variables such as the velocity v) used in the DSMC algorithms, for the selection of particle pairs and collision parameters, as well as for the rejection sampling step. These expectations assume particle velocities $\mathcal{V}^k$ at time t_k are known.

3.1.2. Passing Velocity Derivatives Through the Expectations

For an arbitrary test function ϕ, the expectation for the rejection sampling process for a single pair of velocities $(v_i^k, v_{i_1}^k)$ can be written as

$$E_{ri}^k[\phi] = \sum_{j=1}^{3} h_{ij}^k \phi_j,$$

where the random variable $\phi = \phi_j$ with probability h_{ij}^k. Since h_{ij}^k depends on $(v_i^k, v_{i_1}^k)$, for both $j = 2$ and $j = 3$, the velocity derivative does not commute with E_{ri}^k, but does commute with $E_{ri'}^k$ for $i' \neq i$ and $i' \neq i_1$. We can calculate the velocity derivative as follows:

$$\frac{\partial}{\partial v_i^k} E_{ri}^k[\phi] = \sum_{j=1}^{3} h_{ij}^k \left(\frac{\partial}{\partial v_i^k} \phi_j + \phi_j \frac{\partial}{\partial v_i^k} \log h_{ij}^k \right) = E_{ri}^k \left[\frac{\partial}{\partial v_i^k} \phi + \phi \frac{\partial}{\partial v_i^k} \log h_i^k \right].$$

In the last term, h_i^k can be considered as a random variable for the velocity pair $(v_i^k, v_{i_1}^k)$, and $h_i^k = h_{ij}^k$ with probability h_{ij}^k. Since $E_r^k = \Pi_{i'=1}^{N/2} E_{ri'}^k$ and $\partial / \partial v_i^k$ commutes with $E_{ri'}^k$ for $i' \neq i$ and $i' \neq i_1$, it follows that

$$\frac{\partial}{\partial v_i^k} E_r^k[\phi] = E_r^k \left[\frac{\partial}{\partial v_i^k} \phi + \phi \left(\frac{\partial}{\partial v_i^k} \log h_i^k \right) \right].$$

Because E_p^k also commutes with velocity derivatives, it follows that

$$\frac{\partial}{\partial v_i^k} E^k[\phi] = \frac{\partial}{\partial v_i^k} E_p^k E_r^k[\phi] = E^k \left[\frac{\partial}{\partial v_i^k} \phi + \phi \left(\frac{\partial}{\partial v_i^k} \log h_i^k \right) \right]. \tag{18}$$

3.1.3. Objective Function

We rewrite the objective function (16) at time $T = t_M$ as

$$J = E\left[\bar{\phi}^M\right], \quad \text{where} \quad \bar{\phi}^M = \frac{\rho}{N} \sum_{i=1}^{N} \phi_i^M, \tag{19}$$

where $\rho = \int f(v,T)dv$, $\phi_i^M = \phi(v_i^M)$ with $v_i^M \sim f(v,T)$ (in which the notation $v \sim f$ means that the random variable v is sampled from the distribution function f). Note that (19) is an average of expectations over the N velocities, rather than a single expectation of v with respect to a density function $f(v,T)$ as defined in (16). The expectation E is taken over all the randomness in the forward DSMC simulation over M time steps.

Next, we denote the conditional average at time t_k as

$$J^k = E\left[\bar{\phi}^M | \mathcal{V}^k\right], \tag{20}$$

which is the expectation of $\bar{\phi}^M$ for known values of the velocities v_i^k at time t_k. Since

$$E\left[\cdot | \mathcal{V}^k\right] = E^k \dots E^{M-1}\left[\cdot | \mathcal{V}^k\right], \tag{21}$$

we then have

$$E\left[\bar{\phi}^M | \mathcal{V}^k\right] = E^k\left[\ E\left[\bar{\phi}^M | \mathcal{V}^{k+1}\right]\ | \mathcal{V}^k\right],$$

which implies

$$J^k = E^k\left[J^{k+1} | \mathcal{V}^k\right], \quad k = 0, \dots, M-1. \tag{22}$$

Note that although the expectations E^k and E^{k+1} are independent and commute since the former is a function of $\mathcal{V}^k$ and the latter is a function of $\mathcal{V}^{k+1}$, their order in (21) is necessary for the interpretation as a conditional expectation and to relate $\mathcal{V}^{k+1}$ with $\mathcal{V}^k$.

If we also define

$$J_i^k = E\left[\phi_i^M | \mathcal{V}^k\right], \tag{23}$$

then by the same reasoning

$$J_i^k = E^k\left[J_i^{k+1} | \mathcal{V}^k\right], \quad k = 0, \dots, M-1.$$

Moreover, for a fixed index $i = 1, \dots, N$, we have the following

$$J_i^k = E[\phi_i^M | \mathcal{V}^k] = E_p^k E_{ri}^k \dots E_p^{M-1} E_{ri}^{M-1}[\phi_i^M | \mathcal{V}^k], \quad k = 0, \dots, M-1.$$

Recall that $\{E_p^k\}$ are expectations over the choice of collision pairs and parameters, and $\{E_{ri}^k\}$ are the expectations over the rejection sampling at each time step for the i-th particle as defined in Section 3.1.1. Each probability h_{ij}^k, $j = 1, 2, 3$, at time t_k for particle index i, depends on v_i^k and its collision partner $v_{i_1}^k$ at time t_k. Thus, for any k,

$$J^k = \frac{\rho}{N} \sum_{i=1}^{N} J_i^k = \frac{\rho}{N} \sum_{i=1}^{N} E[\phi_i^M | \mathcal{V}^k] = \frac{\rho}{N} \sum_{i=1}^{N} E_p^k E_{ri}^k \dots E_p^{M-1} E_{ri}^{M-1}[\phi_i^M | \mathcal{V}^k], \tag{24}$$

which is another way to represent (20).

Note that in (17), the index i ranges from 1 to $N/2$ as we treat (i, i_1) as a pair in the rejection sampling process. In (24), the index for the particle ranges from 1 to N because we need to treat the i-th and the i_1-th particles separately, since the objective function (19) is a sum of N terms. Thus, in (24), we should understand $E_p^k E_{ri_1}^k = E_p^k E_{ri}^k$ if (i, i_1) is a collision pair at time t_k where i ranges from 1 to $N/2$ as in (17).

3.1.4. Influence Function

We introduce the influence function $\gamma_i^k = \gamma_i^k(\mathcal{V}^k)$ defined as

$$\gamma_i^k := \frac{\partial}{\partial v_i^k} J^k = \frac{\partial}{\partial v_i^k} E^k[J^{k+1}|\mathcal{V}^k] = \frac{\rho}{N} \sum_{i'=1}^{N} \frac{\partial}{\partial v_i^k} E^k[J_{i'}^{k+1}|\mathcal{V}^k], \tag{25}$$

where the second equality is a result of (22), and third equation follows (24). After applying (18), we have

$$\frac{\partial}{\partial v_i^k} J_{i'}^k = \frac{\partial}{\partial v_i^k} E^k[J_{i'}^{k+1}|\mathcal{V}^k] = E^k\left[\frac{\partial}{\partial v_i^k} J_{i'}^{k+1} + J_{i'}^{k+1}\left(\frac{\partial}{\partial v_i^k}\log h_{i'}^k\right)\Big|\mathcal{V}^k\right], \tag{26}$$

for $i' = 1, \ldots, N$. Note that $h_{i'j}^k = h_{i_1'j}^k$ if (i', i_1') is a pair at t_k, for $j = 1, 2, 3$. Based on the forward DSMC algorithm, the probabilities in the rejection sampling step, $h_{i'}^k$ in $\{J_{i'}^k\}$ do not depend on v_i^k if $i' \neq i, i_1$. Thus, the last term in (26) disappears for $i' \neq i, i_1$. After calculating the sum in (25), we have

$$\gamma_i^k = E^k\left[\frac{\partial}{\partial v_i^k} J^{k+1} + \frac{\rho}{N} J_i^{k+1}\left(\frac{\partial}{\partial v_i^k}\log h_i^k\right) + \frac{\rho}{N} J_{i_1}^{k+1}\left(\frac{\partial}{\partial v_i^k}\log h_i^k\right)\Big|\mathcal{V}^k\right], \tag{27}$$

where (i, i_1) is a pair at $t = t_k$ in the forward DSMC.

Note that J^k is a function of $\mathcal{V}^k = \{v_i^k\}$ and J^{k+1} is a function of $\mathcal{V}^{k+1} = \{v_i^{k+1}\}$. The only elements in $\mathcal{V}^{k+1}$ that depend on v_i^k are v_i^{k+1} and $v_{i_1}^{k+1}$, which are determined by

$$\begin{pmatrix} v_i^{k+1} \\ v_{i_1}^{k+1} \end{pmatrix} = \mathcal{C}_i^k \begin{pmatrix} v_i^k \\ v_{i_1}^k \end{pmatrix}, \tag{28}$$

where the operator $\mathcal{C}_i^k$ is one of the following three possible matrices each with probability h_{ij}^k, $j = 1, 2, 3$; see (5) for the definition of $A(\sigma, \alpha)$ and Section 3.1.2 for details about $\{h_{ij}^k\}$.

$$\mathcal{C}_i^k = \begin{cases} I, & j = 1, (v_i^k, v_{i_1}^k) \text{ does not have a real or virtual collision,} \\ A(\sigma_i^k, \alpha_i^k), & j = 2, (v_i^k, v_{i_1}^k) \text{ has a real collision,} \\ I, & j = 3, (v_i^k, v_{i_1}^k) \text{ has a virtual, but not a real, collision,} \end{cases} \tag{29}$$

where $I \in \mathbb{R}^6$ is the identity matrix. Note that, in (28), v_i^k and $v_{i_1}^k$ are a collision pair, but that v_i^{k+1} and $v_{i_1}^{k+1}$ are not a collision pair.

Furthermore, based on the definition of C in (6), we have

$$D = \left[\frac{\partial(v', v_1')}{\partial(v, v_1)}\right]^\top = \begin{cases} I, & j = 1, \\ [C(\sigma, \alpha)]^\top, & j = 2, \\ I, & j = 3. \end{cases} \tag{30}$$

We remark here that $A(\sigma,\alpha)$ and $C(\sigma,\alpha)$ may not be the same. We will discuss the concrete form of D in detail in Section 3.3. It follows that

$$\begin{pmatrix}\partial/\partial v_i^k\\ \partial/\partial v_{i_1}^k\end{pmatrix} = \frac{\partial(v_i^{k+1}, v_{i_1}^{k+1})}{\partial(v_i^k, v_{i_1}^k)} \cdot \begin{pmatrix}\partial/\partial v_i^{k+1}\\ \partial/\partial v_{i_1}^{k+1}\end{pmatrix} = D_i^k \begin{pmatrix}\partial/\partial v_i^{k+1}\\ \partial/\partial v_{i_1}^{k+1}\end{pmatrix}, \tag{31}$$

where D_i^k is the matrix D defined in (30) applied to the pair $(v_i^k, v_{i_1}^k)$.

Now we can calculate the change in the influence function over a single step from t_k to t_{k+1}. Similar to (27), we get $\gamma_{i_1}^k$ where

$$\gamma_{i_1}^k = E^k\left[\frac{\partial}{\partial v_{i_1}^k} J^{k+1} + \frac{\rho}{N} J_{i_1}^{k+1}\left(\frac{\partial}{\partial v_{i_1}^k}\log h_i^k\right) + \frac{\rho}{N} J_i^{k+1}\left(\frac{\partial}{\partial v_{i_1}^k}\log h_i^k\right)\Big|\mathcal{V}^k\right]. \tag{32}$$

Combining (27) and (32), and applying (31), we have

$$\begin{pmatrix}\gamma_i^k\\ \gamma_{i_1}^k\end{pmatrix} = E^k\left[D_i^k\begin{pmatrix}\gamma_i^{k+1}\\ \gamma_{i_1}^{k+1}\end{pmatrix}\Big|\mathcal{V}^k\right] + \frac{\rho}{N}E^k\left[\left(J_i^{k+1} + J_{i_1}^{k+1}\right)\begin{pmatrix}\partial \log h_i^k/\partial v_i^k\\ \partial \log h_i^k/\partial v_{i_1}^k\end{pmatrix}\Big|\mathcal{V}^k\right]. \tag{33}$$

Again, h_i^k in the right-hand side of (33) should be seen as a random variable, and $h_i^k = h_{ij}^k = h_j$ with probability h_j where $\{h_j\}$ is defined in Section 2.2.

Now we apply sampling to this expectation (33), as in the forward DSMC algorithm, over a single sample of the dynamics of N discrete particles. We remark that based on (23), J_i^{k+1} and $J_{i_1}^{k+1}$ become ϕ_i^M and $\phi_{i_1}^M$ after sampling. Finally, we obtain

$$\begin{pmatrix}\gamma_i^k\\ \gamma_{i_1}^k\end{pmatrix} = D_i^k\begin{pmatrix}\gamma_i^{k+1}\\ \gamma_{i_1}^{k+1}\end{pmatrix} + \frac{\rho}{N}\left(\phi_i^M + \phi_{i_1}^M\right)\frac{\partial \log h_i^k}{\partial v_i^k}\begin{pmatrix}1\\ -1\end{pmatrix}, \tag{34}$$

because $\frac{\partial}{\partial v_i^k}\log h_i^k = -\frac{\partial}{\partial v_{i_1}^k}\log h_i^k$ due to the symmetry of elastic binary collision.

Note that the velocity derivative (an optimization step) is applied to the expectation in (24) and then sampling (a discretization step) is applied to the result in (34). As discussed in Section 1, this part of the derivation follows the OTD approach. On the other hand, instead of using a probability density function f as in (16), the expectation in (24) is based on N discrete velocity particles (using particle discretization). Thus, this part follows the DTO approach. The derivation here is a combination of both DTO and OTD approaches.

3.1.5. Final Data

The "final data" for γ_i^k with $k = M$ is simply

$$\gamma_i^M = \frac{\partial}{\partial v_i^M} J^M = \frac{\partial}{\partial v_i^M} E[\bar{\phi}^M|\mathcal{V}^M] = \frac{\partial}{\partial v_i^M}\bar{\phi}^M = \frac{\rho}{N}\phi'(v_i^M). \tag{35}$$

3.1.6. Resulting Adjoint System

Combining both (34) and (35), the resulting system is

$$\gamma_i^M = \frac{\rho}{N}\phi'(v_i^M) \tag{36a}$$

$$\begin{pmatrix}\gamma_i^k\\ \gamma_{i_1}^k\end{pmatrix} = D_i^k\begin{pmatrix}\gamma_i^{k+1}\\ \gamma_{i_1}^{k+1}\end{pmatrix} + \begin{pmatrix}\eta_i^k\\ \eta_{i_1}^k\end{pmatrix} \quad \text{for } k = 0,\dots,M-1 \tag{36b}$$

in which η_i^k is defined as

$$\eta_i^k = \begin{cases} 0, & \text{if } v_i^k \text{ does not have a virtual or real collision,} \\ \dfrac{\rho}{N}\dfrac{\partial \log q_i^k}{\partial v_i^k}(\phi_i^M + \phi_{i_1}^M), & \text{if } v_i^k \text{ has a virtual collision that is a real collision,} \\ \dfrac{\rho}{N}\dfrac{\partial \log(\Sigma - q_i^k)}{\partial v_i^k}(\phi_i^M + \phi_{i_1}^M), & \text{if } v_i^k \text{ has a virtual, but not a real, collision.} \end{cases} \tag{37}$$

This can be used to calculate sensitivities. Assuming the L^2 inner product for m, we have

$$\nabla_m J = \nabla_m \left(E_{\mathcal{V}^0} \left[E \left[\bar{\phi}^M | \mathcal{V}^0 \right] \right] \right) \approx \sum_{i=1}^N \frac{\partial v_i^0}{\partial m} \cdot \left(\frac{\partial}{\partial v_i^0} E \left[\bar{\phi}^M | \mathcal{V}^0 \right] \right) \approx \sum_{i=1}^N \frac{\partial v_i^0}{\partial m} \cdot \gamma_i^0.$$

We denote the gradient calculated through the adjoint approach as $\nabla_m J^{AD}$ where

$$\nabla_m J^{AD} = \sum_{i=1}^N \nabla_m v_i^0 \cdot \gamma_i^0. \tag{38}$$

Note that $E_{\mathcal{V}^0}$ denotes the expectation over all elements in $\mathcal{V}^0$, and $\forall v_i^0 \in \mathcal{V}^0$, $v_i^0 \sim f_0(v; m)$, the initial distribution introduced in (15). The approximation in (38) corresponds to the so-called pathwise gradient estimator [16, Section 5].

To compute the gradient represented by the last term in (38), we need to "back-propagate" (36) from $t = t_M = T$ all the way to $t = t_0 = 0$. We remark that the only differences between (36) and the adjoint DSMC algorithm designed for the Maxwell molecules [9] are the "η" terms in (36b). We summarize the new adjoint DSMC algorithm in Algorithm 2.

Algorithm 2 Solve the Discrete Adjoint DSMC System and Compute the Gradient (38).

1: Given the final-time velocities $\mathcal{V}_M$ from the forward DSMC, set $\gamma_i^M = \frac{\rho}{N}\phi'(v_i^M)$ for all i.
2: **for** $k = M - 1$ to 0 **do**
3: Given $\{\gamma_1^{k+1}, \ldots, \gamma_N^{k+1}\}$ from the previous iteration, the collision history and collision parameters from the forward DSMC process.
4: **if** $v_i^k \in \mathcal{V}_k$ did not virtually collide at t_k **then**
5: Set $\gamma_i^k = \gamma_i^{k+1}$.
6: **else if** $v_i^k, v_{i_1}^k \in \mathcal{V}_k$ virtually collided at t_k **then**
7: Set $\begin{pmatrix} \gamma_i^k \\ \gamma_{i_1}^k \end{pmatrix} = D_i^k \begin{pmatrix} \gamma_i^{k+1} \\ \gamma_{i_1}^{k+1} \end{pmatrix} + \begin{pmatrix} \eta_i^k \\ \eta_{i_1}^k \end{pmatrix}$, where $\eta_i^k, \eta_{i_1}^k$ follow (37).
8: **end if**
9: **end for**
10: Compute the gradient following (38).

3.2. A Lagrangian Approach

In addition to the direct derivation of the adjoint equations described above, it is useful (as mentioned in the Introduction) to have a "Langrangian" form from which the forward DSMC and the adjoint equations can be derived. Using the earlier notation E^k in Section 3.1 and the objective function J defined in (19), we can write the Lagrangian $\mathcal{J}$ as

$$\begin{aligned}\mathcal{J} &= J + \frac{1}{2}\sum_{k=0}^{M-1}\sum_{i=1}^{N} E^k\left[\begin{pmatrix}\gamma_i^{k+1}\\ \gamma_{i_1}^{k+1}\end{pmatrix}\cdot\left(\mathcal{C}_i^k\begin{pmatrix}v_i^k\\ v_{i_1}^k\end{pmatrix} - \begin{pmatrix}v_i^{k+1}\\ v_{i_1}^{k+1}\end{pmatrix}\right)\middle|\begin{pmatrix}v_i^k\\ v_{i_1}^k\end{pmatrix}\right] + \\ &\quad \sum_{i=1}^{N} E_{\hat{v}_{0i}(m)\sim f_0(v;m)}\left[\gamma_i^0\cdot\left(\hat{v}_{0i}(m) - v_i^0\right)\right]. \end{aligned} \tag{39}$$

The "1/2" scaling is to avoid enforcing the collision rule twice. Again, note that v_i^k and $v_{i_1}^k$ are a collision pair, but that neither $(v_i^{k+1}, v_{i_1}^{k+1})$ nor $(v_i^M, v_{i_1}^M)$ are a collision pair.

In contrast to the direct approach in which the velocities $\{v_i^k\}$ are chosen according to the forward DSMC and the $\{\gamma_i^k\}$ are chosen by the adjoint equations, in the Lagrangian approach described here, the velocities $\{v_i^k\}$ and the Lagrangian multipliers $\{\gamma_i^k\}$ are considered to be any random variables. We then derive the forward and adjoint DSMC equations by requiring that $\mathcal{J}$ is stationary with respect to variations in the v_i^k's and the γ_i^k's.

3.2.1. Collision Rules and Initial Data

The collision rules are derived from the derivatives of $\mathcal{J}$ with respect to γ_i^{k+1} and $\gamma_{i_1}^{k+1}$ for $k = 0, \ldots, M-1$. Setting these to zero implies that

$$\mathcal{C}_i^k\begin{pmatrix}v_i^k\\ v_{i_1}^k\end{pmatrix} = \begin{pmatrix}v_i^{k+1}\\ v_{i_1}^{k+1}\end{pmatrix}. \tag{40}$$

See (29) for the three possible cases of $\mathcal{C}_i^k$, which are the DSMC equations for a non-virtual collision, a real collision and a virtual, but not real, collision, respectively.

Similarly, setting to zero the derivatives of $\mathcal{J}$ with respect to γ_i^0 implies that

$$v_i^0 = \hat{v}_{0i}(m),$$

which is the initial data for the forward DSMC. We assume that $\hat{v}_{0i}(m) \sim f_0(v; m)$.

3.2.2. Parameter m

If we take the derivative of $\mathcal{J}$ with respect to the parameter m, we obtain the gradient based on sampled $\{\hat{v}_{0i}\}$:

$$\frac{\partial J}{\partial m} = \frac{\partial \mathcal{J}}{\partial m} \approx \sum_{i=1}^{N}\gamma_i^0\cdot\frac{\partial \hat{v}_{0i}(m)}{\partial m}. \tag{41}$$

Note that (41) is the same as the $\nabla_m J^{AD}$ in (38).

3.2.3. Final Data

For each i, we take the derivative of $\mathcal{J}$ with respect to the final velocity particle v_i^M. There are three terms in $\mathcal{J}$ defined in (39). For the first term J, we consider $J \approx J^M = E[\bar{\phi}^M|\mathcal{V}^M] = \rho N^{-1}\sum_{i'=1}^N \phi(v_{i'}^M)$ as defined in (19) and (20). For the second term which enforces the binary collision rule, since the expectation E^{M-1} does not depend on particle velocities in $\mathcal{V}^M$, we have

$$\frac{\partial}{\partial v_i^M} E^{M-1}[\gamma_i^M \cdot v_i^M|\mathcal{V}^{M-1}] = E^{M-1}[\gamma_i^M|\mathcal{V}^{M-1}].$$

The last term in $\mathcal{J}$ does not depend on the final-time particles $\mathcal{V}^M$, so does not contribute to the derivative. Summarizing all the terms, we have

$$\begin{aligned}\frac{\partial \mathcal{J}}{\partial v_i^M} &\approx \frac{\partial}{\partial v_i^M}\left(\rho N^{-1}\sum_{i'=1}^N \phi(v_{i'}^M) - E^{M-1}\left[\gamma_i^M \cdot v_i^M|\mathcal{V}^{M-1}\right]\right)\\ &= \rho N^{-1}\phi'(v_i^M) - E^{M-1}\left[\gamma_i^M|\mathcal{V}^{M-1}\right] \approx \rho N^{-1}\phi'(v_i^M) - \gamma_i^M.\end{aligned}$$

Setting this derivative to 0 implies the final condition $\gamma_i^M = \rho N^{-1}\phi'(v_i^M)$ as seen in (36a).

3.2.4. Adjoint Equations

The first term J in the Lagrangian is defined as an expectation of ϕ^M; see (19). For the first term J in the Lagrangian, we consider $J \approx J^k = E\left[\bar{\phi}^M|\mathcal{V}^k\right]$ since we are taking the derivative of J with respect to elements in $\mathcal{V}^k$. That is,

$$J \approx J^k = E\left[\phi^M|\mathcal{V}^k\right] = \rho N^{-1}\sum_{i'=1}^N E^k \dots E^{M-1}[\phi_{i'}^M].$$

Since v_i^k only appears in the terms $h_{ij}^k = h_j(v_i^k - v_{i_1}^k, \sigma_i^k)$, $j = 1, 2, 3$, in E^k (see Section 2.2), then using the commutator between the derivative and the expectation as in (18), we have

$$\begin{aligned}\partial_{v_i^k} J &\approx \rho N^{-1}\sum_{i'=1}^N (\partial_{v_i^k} E^k) E^{k+1} \dots E^{M-1}[\phi_{i'}^M]\\ &= \rho N^{-1}\sum_{i'=1}^N E^k \dots E^{M-1}\left[(\delta_{i'i} + \delta_{i'i_1})\left(\partial_{v_i^k}\log h_i^k\right)\phi_{i'}^M\right]\\ &= \rho N^{-1} E^k \dots E^{M-1}\left[\partial_{v_i^k}\log h_i^k\left(\phi_i^M + \phi_{i_1}^M\right)\right]\\ &\approx \rho N^{-1}\left(\partial_{v_i^k}\log h_i^k\right)[\phi_i^M + \phi_{i_1}^M], \end{aligned} \tag{42}$$

in which the approximation in the last step is from sampling. The h_i^k term in all equations above except the last one should be considered as a random variable and $h_i^k = h_{ij}^k$ with probability h_{ij}^k, $j = 1, 2, 3$. We use the final term in (42) as the contribution from J to the (approximate) optimality condition for $\mathcal{J}$.

We continue by differentiating the remaining term $\mathcal{J} - J$ in (39) to obtain

$$\begin{aligned}\frac{\partial(\mathcal{J}-J)}{\partial v_i^k} &= E^k\left[\begin{pmatrix}\gamma_i^{k+1}\\ \gamma_{i_1}^{k+1}\end{pmatrix}\cdot\left\{\mathcal{C}_i^k\begin{pmatrix}v_i^k\\ v_{i_1}^k\end{pmatrix}-\begin{pmatrix}v_i^{k+1}\\ v_{i_1}^{k+1}\end{pmatrix}\right\}\frac{\partial\log h_i^k}{\partial v_i^k}\middle|\mathcal{V}^k\right]+\\ &\quad E^k\left[\overline{D}_i^k\begin{pmatrix}\gamma_i^{k+1}\\ \gamma_{i_1}^{k+1}\end{pmatrix}\middle|\mathcal{V}^k\right]-E^{k-1}\left[\gamma_i^k|\mathcal{V}^{k-1}\right],\\ &\approx \begin{pmatrix}\gamma_i^{k+1}\\ \gamma_{i_1}^{k+1}\end{pmatrix}\cdot\left\{\mathcal{C}_i^k\begin{pmatrix}v_i^k\\ v_{i_1}^k\end{pmatrix}-\begin{pmatrix}v_i^{k+1}\\ v_{i_1}^{k+1}\end{pmatrix}\right\}\frac{\partial\log h_i^k}{\partial v_i^k}+\overline{D}_i^k\begin{pmatrix}\gamma_i^{k+1}\\ \gamma_{i_1}^{k+1}\end{pmatrix}-\gamma_i^k, \qquad (43)\end{aligned}$$

$$\begin{aligned}\frac{\partial(\mathcal{J}-J)}{\partial v_{i_1}^k} &= E^k\left[\begin{pmatrix}\gamma_i^{k+1}\\ \gamma_{i_1}^{k+1}\end{pmatrix}\cdot\left\{\mathcal{C}_i^k\begin{pmatrix}v_i^k\\ v_{i_1}^k\end{pmatrix}-\begin{pmatrix}v_i^{k+1}\\ v_{i_1}^{k+1}\end{pmatrix}\right\}\frac{\partial\log h_i^k}{\partial v_{i_1}^k}\middle|\mathcal{V}^k\right]+\\ &\quad E^k\left[\underline{D}_i^k\begin{pmatrix}\gamma_i^{k+1}\\ \gamma_{i_1}^{k+1}\end{pmatrix}\middle|\mathcal{V}^k\right]-E^{k-1}\left[\gamma_{i_1}^k|\mathcal{V}^{k-1}\right]\\ &\approx \begin{pmatrix}\gamma_i^{k+1}\\ \gamma_{i_1}^{k+1}\end{pmatrix}\cdot\left\{\mathcal{C}_i^k\begin{pmatrix}v_i^k\\ v_{i_1}^k\end{pmatrix}-\begin{pmatrix}v_i^{k+1}\\ v_{i_1}^{k+1}\end{pmatrix}\right\}\frac{\partial\log h_i^k}{\partial v_{i_1}^k}+\underline{D}_i^k\begin{pmatrix}\gamma_i^{k+1}\\ \gamma_{i_1}^{k+1}\end{pmatrix}-\gamma_{i_1}^k. \qquad (44)\end{aligned}$$

Here, $D_i^k = \begin{bmatrix}\overline{D}_i^k\\ \underline{D}_i^k\end{bmatrix}$. That is, $\overline{D}_i^k, \underline{D}_i^k \in \mathbb{R}^{3\times 6}$ are the upper and lower blocks of the matrix D_i^k defined in (30). Note that the terms in brackets "{ }" are always 0 because of (40).

Finally, we combine (42), (43) and (44), to obtain approximations for $\frac{\partial\mathcal{J}}{\partial v_i^k}$ and $\frac{\partial\mathcal{J}}{\partial v_{i_1}^k}$. Setting them to zero results in equations that are identical to (36); i.e., the Lagrangian approach confirms the results of the direct approach.

Note that the two derivations of adjoint equations, the first directly from the DSMC equations (Section 3.1) and the second from a Lagrangian (Section 3.2), are nearly identical in terms of the details of the calculations and the origin of the score function.

3.3. The Calculation of D in (30)

In the earlier adjoint DSMC derivations, we did not specify the explicit form of the matrix D in (30), which is denoted as D_i^k when applied to a particular collision pair $(v_i^k, v_{i_1}^k)$. It is clear that the matrix is an identity matrix when the velocity particles do not have a real collision. Thus, we focus on the case $j = 2$.

Based on the binary collision formula (2), there are two cases when we calculate the Jacobian matrix C and its adjoint D: (1) σ can be seen to be independent of (v, v_1) when the collision kernel $q(v - v_1, \sigma)$ is angle-independent, and (2) σ depends on (v, v_1) otherwise.

Case one – angle-independent kernel: Despite the fact that σ is the unit vector along the post-collision relative velocity, σ can be regarded as uniformly distributed over $\mathcal{S}^2$ as long as the collision kernel $q(v - v_1, \sigma) = \tilde{q}(|v - v_1|, \theta)$ does not depend on the scattering angle θ. Thus, we can consider σ to be independent of the pre-collision relative velocity α, and thus independent of (v, v_1). If (v, v_1) is a real collision pair, based on (4), we have

$$D = C(\sigma, \alpha)^\top = \left[\frac{\partial(v', v_1')}{\partial(v, v_1)}\right]^\top = B(\sigma, \alpha), \qquad (45)$$

where B is defined in (5). This formula was used in [9] where a constant collision kernel was considered.

Case two – angle-dependent kernel: In this case, we need to consider the dependence of σ on the pre-collision particle velocities, v and v_1, since the distribution of σ is no longer uniform over the sphere $\mathcal{S}^2$, and it depends on $\alpha = \frac{v-v_1}{|v-v_1|}$. Going through the calculations for (6), we have

$$\begin{aligned}
\partial_v v' &= \frac{1}{2}\left(I + \sigma\ \alpha^\top + |u|\partial_v\sigma\right), \\
\partial_{v_1} v' &= \frac{1}{2}\left(I - \sigma\ \alpha^\top + |u|\partial_{v_1}\sigma\right), \\
\partial_v v_1' &= \frac{1}{2}\left(I - \sigma\ \alpha^\top - |u|\partial_v\sigma\right), \\
\partial_{v_1} v_1' &= \frac{1}{2}\left(I + \sigma\ \alpha^\top - |u|\partial_{v_1}\sigma\right),
\end{aligned}$$

where $u = v - v_1$. Note that $\partial_v\sigma = \partial_u\sigma = -\partial_{v_1}\sigma$. Next, we compute $\partial_v\sigma$ and $\partial_{v_1}\sigma$. Note that we can also write the collision formula (2), following [24, Sec. 2.2], as

$$\begin{aligned}
v' &= v + 1/2\,\Delta U, \\
v_1' &= v_1 - 1/2\,\Delta U,
\end{aligned}$$

where

$$\Delta U = \begin{pmatrix} \frac{u_x u_z}{\sqrt{u_x^2+u_y^2}} & -\frac{u_y|u|}{\sqrt{u_x^2+u_y^2}} & u_x \\ \frac{u_y u_z}{\sqrt{u_x^2+u_y^2}} & \frac{u_x|u|}{\sqrt{u_x^2+u_y^2}} & u_y \\ -\sqrt{u_x^2+u_y^2} & 0 & u_z \end{pmatrix} \begin{pmatrix} \sin\theta\cos\varphi \\ \sin\theta\sin\varphi \\ \cos\varphi \end{pmatrix} - u.$$

Thus, using the definition of σ, we have

$$\sigma = \frac{v'-v_1'}{|v'-v_1'|} = \frac{v'-v_1'}{|v-v_1|} = \begin{pmatrix} \frac{u_x u_z}{|u|\sqrt{u_x^2+u_y^2}} & -\frac{u_y}{\sqrt{u_x^2+u_y^2}} & \frac{u_x}{|u|} \\ \frac{u_y u_z}{|u|\sqrt{u_x^2+u_y^2}} & \frac{u_x}{\sqrt{u_x^2+u_y^2}} & \frac{u_y}{|u|} \\ -\frac{\sqrt{u_x^2+u_y^2}}{|u|} & 0 & \frac{u_z}{|u|} \end{pmatrix} \begin{pmatrix} \sin\theta\cos\varphi \\ \sin\theta\sin\varphi \\ \cos\varphi \end{pmatrix}, \tag{46}$$

which shows the dependence of σ upon v, v_1, θ, φ explicitly. For fixed θ and φ, we have

$$\begin{aligned}
|u|\partial_u\sigma &= |u|\left(\frac{\partial e_i^\top \sigma}{\partial e_j^\top u}\right)_{ij} = \begin{bmatrix} G_1\sigma & G_2\sigma & G_3\sigma \end{bmatrix}, \\
|u|\left(\partial_u\sigma\right)^\top &= \begin{bmatrix} \sigma^\top G_1^\top \\ \sigma^\top G_2^\top \\ \sigma^\top G_3^\top \end{bmatrix} = -\begin{bmatrix} \sigma^\top G_1 \\ \sigma^\top G_2 \\ \sigma^\top G_3 \end{bmatrix},
\end{aligned}$$

where $\{e_i\}$ is the standard basis in $\mathbb{R}^3$, and matrices $G_j = -G_j^\top$, $j = 1, 2, 3$, as defined below

using the notation $\alpha = u/|u| = [\alpha_x, \alpha_y, \alpha_z]^\top$.

$$G_1 = \begin{bmatrix} 0 & \frac{\alpha_y}{\alpha_x^2+\alpha_y^2} & \frac{\alpha_x^2\alpha_z}{\alpha_x^2+\alpha_y^2} \\ -\frac{\alpha_y}{\alpha_x^2+\alpha_y^2} & 0 & \frac{\alpha_x\alpha_y\alpha_z}{\alpha_x^2+\alpha_y^2} \\ -\frac{\alpha_x^2\alpha_z}{\alpha_x^2+\alpha_y^2} & -\frac{\alpha_x\alpha_y\alpha_z}{\alpha_x^2+\alpha_y^2} & 0 \end{bmatrix} = \frac{1}{\alpha_x^2+\alpha_y^2} \begin{bmatrix} 0 & \alpha_y & \alpha_x^2\alpha_z \\ -\alpha_y & 0 & \alpha_x\alpha_y\alpha_z \\ -\alpha_x^2\alpha_z & -\alpha_x\alpha_y\alpha_z & 0 \end{bmatrix},$$

$$G_2 = \begin{bmatrix} 0 & -\frac{\alpha_x}{\alpha_x^2+\alpha_y^2} & \frac{\alpha_x\alpha_y\alpha_z}{\alpha_x^2+\alpha_y^2} \\ \frac{\alpha_x}{\alpha_x^2+\alpha_y^2} & 0 & \frac{\alpha_y^2\alpha_z}{\alpha_x^2+\alpha_y^2} \\ -\frac{\alpha_x\alpha_y\alpha_z}{\alpha_x^2+\alpha_y^2} & -\frac{\alpha_y^2\alpha_z}{\alpha_x^2+\alpha_y^2} & 0 \end{bmatrix} = \frac{1}{\alpha_x^2+\alpha_y^2} \begin{bmatrix} 0 & -\alpha_x & \alpha_x\alpha_y\alpha_z \\ \alpha_x & 0 & \alpha_y^2\alpha_z \\ -\alpha_x\alpha_y\alpha_z & -\alpha_y^2\alpha_z & 0 \end{bmatrix},$$

$$G_3 = \begin{bmatrix} 0 & 0 & -\alpha_x \\ 0 & 0 & -\alpha_y \\ \alpha_x & \alpha_y & 0 \end{bmatrix}.$$

We define a tensor G_{lij} where $G_{1ij} = G_1$, $G_{2ij} = G_2$ and $G_{3ij} = G_3$. We will then use Einstein's summation convention. The adjoint matrix D in (30) for a real collision pair becomes

$$\begin{aligned} D = [C(\sigma,\alpha)]^\top &= \frac{1}{2}\begin{pmatrix} I+\alpha\sigma^\top & I-\alpha\sigma^\top \\ I-\alpha\sigma^\top & I+\alpha\sigma^\top \end{pmatrix} + \frac{1}{2}\begin{pmatrix} -(\sigma)_i G_{lij} & (\sigma)_i G_{lij} \\ (\sigma)_i G_{lij} & -(\sigma)_i G_{lij} \end{pmatrix} \\ &= B(\sigma,\alpha) + \widetilde{B}(\sigma,\alpha), \end{aligned} \tag{47}$$

where $(\sigma)_i$ denotes the i-th element of vector σ, and

$$\widetilde{B}(\sigma,\alpha) = \frac{1}{2}\begin{pmatrix} -(\sigma)_i G_{lij} & (\sigma)_i G_{lij} \\ (\sigma)_i G_{lij} & -(\sigma)_i G_{lij} \end{pmatrix}. \tag{48}$$

In order to build $\widetilde{B}(\sigma,\alpha)$, we need σ and α, which are already stored in memory during the forward DSMC for the calculation of $B(\sigma,\alpha)$ in (5). Therefore, there is no additional memory requirement.

3.4. Special Cases

In this section, we discuss a few special cases of the collision kernel $q(v-v_1,\sigma)$ that may result in a variation or simplification with respect to Algorithm 2.

3.4.1. The DSMC and the Adjoint DSMC Methods for a Constant Kernel

For a constant collision kernel q, which is the model for Maxwell molecules, the upper bound can be taken as $\Sigma = q$. Then $\{h_j\}$ in Section 2.2 are all constant, all virtual collisions are real collisions, and all η terms are 0 in (37). This corresponds to the case studied in [9].

Algorithm 3 DSMC Algorithm for Collision Kernel of Form (8)

Compute the initial velocity particles based on the initial condition, $\mathcal{V}^0 = \{v_1^0, \ldots, v_N^0\}$. Set $N_c = \lceil N\Delta t \mu_\kappa / 2 \rceil$ where $\mu_\kappa = A_\kappa \Sigma_v \rho$ and $M = T/\Delta t$ given final time T.
for $k = 0$ to $M - 1$ **do**
    Given $\mathcal{V}^k$, choose N_c virtual collision pairs (i_ℓ, i_{ℓ_1}) uniformly without replacement. The remaining $N - 2N_c$ particles do not have a virtual (or real) collision and set $v_i^{k+1} = v_i^k$.
    for $\ell = 1$ to N_c **do**
        Compute $u_\ell = |v_{i_\ell}^k - v_{i_{\ell_1}}^k|^\beta$.
        Draw a random number ξ_ℓ from the uniform distribution $\mathcal{U}([0, 1])$.
        if $\xi_\ell \le u_\ell / \Sigma_v$ **then**
            Sample scattering angle $\theta \sim C_\kappa(\theta)\sin(\theta)$ and azimuthal angle $\varphi \sim \mathcal{U}([0, 2\pi])$.
            Perform real collision between $v_{i_\ell}^k$ and $v_{i_{\ell_1}}^k$ following (2) and obtain $(v_{i_\ell}^{k\prime}, v_{i_{\ell_1}}^{k\prime})$.
            Set $(v_{i_\ell}^{k+1}, v_{i_{\ell_1}}^{k+1}) = (v_{i_\ell}^{k\prime}, v_{i_{\ell_1}}^{k\prime})$.
        else
            The virtual collision is not a real collision. Set $(v_{i_\ell}^{k+1}, v_{i_{\ell_1}}^{k+1}) = (v_{i_\ell}^k, v_{i_{\ell_1}}^k)$.
        end if
    end for
end for

3.4.2. The DSMC and the Adjoint DSMC Methods for Kernel of Type (8)

The DSMC method presented in Algorithm 1 applies to any general collision kernel $q(v - v_1, \sigma)$ with an upper bound Σ. However, many common collision models for the Boltzmann equation are of the form (8), for which the collision kernel contains two separate parts: the relative velocity part $|v - v_1|^\beta$, and the scattering angle-dependent part $C_\kappa(\theta)$. Given this separation of variables, the DSMC method presented in Algorithm 1 can be modified to improve sampling efficiency.

In contrast to the upper bound Σ for the entire collision kernel as defined in (7), we define Σ_v as the upper bound for the relative velocity part where

$$|v - v_1|^\beta < \Sigma_v. \tag{49}$$

Numerically, the upper bound can be taken as $\Sigma_v = \max_i |v_i - v_{i_1}|^\beta$ where i is the index of a velocity particle and i_1 is its collision partner index [19]. Besides, we define the weighted surface area

$$A_\kappa = \int_0^{2\pi} \int_0^\pi C_\kappa(\theta)\sin(\theta) d\theta d\varphi, \tag{50}$$

which is different from the unweighted surface area A defined in (10). Correspondingly, we have a different collision rate $\mu_\kappa = A_\kappa \Sigma_v \rho$ where the density $\rho = \int f dv$. We summarize the modified DSMC algorithm in Algorithm 3.

Compared with Algorithm 1, there is a major change in Algorithm 3. The collision scattering angle θ and the azimuthal angle φ are sampled independently and directly instead of using the rejection sampling. Since $\theta \in \mathbb{R}$, it is quite efficient to use inverse transform sampling to sample $C_\kappa(\theta)\sin(\theta)$.

Note that Algorithm 3 can be used to sample angle-independent kernels, i.e., $q(v-v_1,\sigma) = |v-v_1|^\beta$. However, since the scattering and azimuthal angles are sampled uniformly first, there is a numerical dependence of the resulting σ on $u = v - v_1$ based on eq. (46), despite that there is no σ-dependence in the collision kernel itself. Technically, the adjoint Jacobian matrix D should follow the formulation (47) in the corresponding adjoint DSMC Algorithm 2 instead of (45). Nevertheless, we observe that the additional term $\widetilde{B}$ in (47) can be dropped due to an averaging effect, reducing it to (45). The resulting gradient is not affected by $\widetilde{B}$. We will further demonstrate it numerically in Section 4.1.

On the other hand, Algorithm 1 does not require sampling the scattering angle and the azimuthal angle first for a general angle-dependent kernel $q(v - v_1, \sigma)$. It samples (v, v_1, σ) independently before going through the rejection sampling step. Although σ is originally sampled independent of (v, v_1), its acceptance probability in the rejection sampling, $q(v - v_1, \sigma)$, enforces the dependence of the final accepted σ on (v, v_1). As a result, the adjoint Jacobian matrix D in the corresponding adjoint DSMC Algorithm 2 should still have the $\widetilde{B}$ term defined in (48) when the forward DSMC method follows Algorithm 1.

4. Numerical Results

In this section, we numerically test the adjoint DSMC method described in Algorithm 2 by computing the gradient using formula (38) for the objective function discussed in Equations (16) and (19),

$$J(m) = \int_{\mathbb{R}^3} \phi(v) f(v,T) dv \approx \bar{\phi}^M = \frac{\rho}{N} \sum_{i=1}^{N} \phi(v_i^M),$$

evaluated at the final time $t = T$, with respect to the parameter m in the initial conditions $f_0(v; m)$ for the general collision kernel in the form of (8),

$$q(v - v_1, \sigma) = C_\kappa(\theta)|v - v_1|^\beta = \frac{1+\kappa}{2^{\kappa+2}\pi\epsilon}(1+\cos\theta)^\kappa |v - v_1|^\beta, \;\; \cos\theta = \sigma \cdot \frac{v - v_1}{|v - v_1|}. \tag{51}$$

We choose this particular form of $C_\kappa(\theta)$ in (51) such that $A_\kappa = 1/\epsilon$ as defined in (50), where ϵ controls the amount of collisions per unit of time. We first consider collision kernels with only velocity dependence ($\kappa = 0$ and $\beta = \{0, 1, 2\}$) and next we consider collision kernels with both velocity and angle dependence ($\kappa = \{1, 2, 5\}$ and $\beta = 1$). We also assume that $\rho(t) = \int_{\mathbb{R}^3} f(v,t)dv = 1$ (which is conserved throughout the evolution by the forward DSMC Algorithms 1 and 3), and thus $\mu_\kappa = A_\kappa \Sigma_v \rho = \Sigma_v/\epsilon$.

Since we do not have an exact solution to the Boltzmann equation (1) under this general collision kernel, we compare the gradient $\nabla_m J^{AD}$ computed by the adjoint DSMC method (via Algorithm 2 and eq. (38)) with the one computed by the central finite difference method,

$$\nabla_m J^{FD}(m) \approx \frac{J(m + \Delta m) - J(m - \Delta m)}{2\Delta m}, \tag{52}$$

where both $J(m + \Delta m)$ and $J(m - \Delta m)$ are computed using forward DSMC simulations with the same random seed. The random seed is used to initialize a pseudorandom number

generator for sampling steps in the forward DSMC algorithms (see Algorithms 1 and 3). We fix the random seed in (52) to reduce the random error in approximating $\nabla_m J^{FD}$. We will also use the average value of $\nabla_m J^{FD}$ from multiple runs (under different random seeds) to further reduce the variance in $\nabla_m J^{FD}$.

We are interested in the error between the two gradients defined as $\left|\nabla_m J^{AD} - \nabla_m J^{FD}\right|$, and studying its convergence as the number of particles N increases in the empirical distribution (14) as a discretization for the distribution function f in the Boltzmann equation (1). This total error has several contributions: the random error from particle discretization in both $\nabla_m J^{AD}$ and $\nabla_m J^{FD}$, and the finite difference error in $\nabla_m J^{FD}$. Thus, when we study the convergence of the total error with N increasing, we expect the total error first to decrease and then to plateau at a fixed constant determined by the finite difference error after the random error is sufficiently small.

To further reduce the random error in both $\nabla_m J^{AD}$ and $\nabla_m J^{FD}$, we perform $M_s = 100$ computations for each of them using random initial conditions, and compute the average values, denoted as $\overline{\nabla_m J}^{AD}$ and $\overline{\nabla_m J}^{FD}$, before computing the error e. That is,

$$e = \left|\overline{\nabla_m J}^{AD} - \overline{\nabla_m J}^{FD}\right|. \tag{53}$$

The random errors in the averaged gradients are estimated by first computing the standard deviation in $\nabla_m J^{AD}$ and $\nabla_m J^{FD}$ respectively using the M_s i.i.d. runs, followed by a rescaling using the factor $1/\sqrt{M_s}$.

Ultimately, we want to show that $|\overline{\nabla_m J}^{AD} - \nabla_m J|$ is small where $\nabla_m J$ is the true gradient. To make $\overline{\nabla_m J}^{FD}$ a better approximation of $\nabla_m J$, one needs to use a tiny perturbation Δm to reduce the central difference error of $\mathcal{O}(|\Delta m|^2)$, and relatively large N and M_s to reduce the random error of $\mathcal{O}((\sqrt{NM_s}|\Delta m|)^{-1})$. In our previous work [9], we balanced the particle discretization (random) error and the finite difference error in $\overline{\nabla_m J}^{FD}$ and found that using $\Delta m = 0.1$ was nearly optimal for $N = 10^6$ Maxwellian particles, $M_s = 100$, and initial conditions similar to (54). Hence, we will also fix $\Delta m = 0.1$ for all simulations here as a heuristic estimation in the following numerical tests.

For the function $\phi(v)$ in (16), we use v_l^2, $l \in \{x, y, z\}$, so the objective functions are

$$T_l = \frac{1}{N}\sum_{i=1}^{N} (v_{F,i}^l)^2 \approx \int_{\mathbb{R}^3} v_l^2 f(v, T) dv, \quad l \in \{x, y, z\},$$

the second-order velocity moments of the distribution function in the l-direction at the time $t = T$. For the parameter m, we use temperature values in the initial distribution function $m = (T_x^0, T_y^0, T_z^0)$. We further refer to these gradients as $\frac{\delta J}{\delta m} = \frac{\partial T_l}{\partial T_p^0}$, $l, p \in \{x, y, z\}$. Here, the dimension of the parameter m is only 3. A real advantage of this adjoint-state method is, of course, when the parameter m is extremely high-dimensional since the adjoint-state method allows one to compute all the components of the gradient $\nabla_m J^{AD}$ by doing only *one* forward DSMC simulation and *one* backward adjoint DSMC simulation.

In all the numerical tests below, we use the same anisotropic Gaussian as the initial condition,

$$f_0(v) = \frac{1}{(2\pi)^{3/2}\sqrt{T_x^0 T_y^0 T_z^0}} \exp\left(-\frac{v_x^2}{2T_x^0} - \frac{v_y^2}{2T_y^0} - \frac{v_z^2}{2T_z^0}\right), \tag{54}$$

where $T_x^0 = 1, T_y^0 = 1, T_z^0 = 0.5$, so that $m = (1, 1, 0.5)$. In this case, the solution to the Boltzmann equation (1) will relax to an isotropic Gaussian with the temperature $T_M = (T_x^0 + T_y^0 + T_z^0)/3 = 0.8333(3)$ as time increases. In the following tests, we use Algorithm 3 for the forward DSMC simulations with a time step $\Delta t = 0.1$ and the final time $t = T = M\Delta t = 2$ at which only partial relaxation to the isotropic Gaussian is attained. Since the initial condition and the solution are anisotropic Gaussians with $T_x^0, T_y^0, T_z^0 \leq 1$, then the upper bound for the relative velocity, $\max_i |v_i - v_{i_1}|$, can be taken to be 10, and consequently we can set $\Sigma_v = 10^\beta$ in (49). Therefore, the fraction of particles that participate in virtual collisions at every time step of the forward DSMC simulations is $N_c/N = \Delta t \mu_\kappa = \Delta t \Sigma_v / \epsilon = \Delta t 10^\beta / \epsilon$, based on Algorithm 3. As discussed in Section 3.4.2, for collision kernels in the form of (8), it is more computationally efficient to use Algorithm 3 as the forward DSMC algorithm than Algorithm 1. In the numerical tests below, we will also only use Algorithm 3, but will comment on the gradient calculation based on Algorithm 1 in Section 4.3.

To compute the gradient $\nabla_m J^{AD}$ using Algorithm 2 and formula (38), we need $\nabla_m v_i^0$. Since our initial distribution (54) is an anisotropic Gaussian, we can sample v_i^0 by sampling $3N$ values from the standard normal distribution $\mathcal{N}(0, 1)$ and then rescaling the values with appropriate initial temperatures as

$$v_i^0 = (v_i^{x,0}, v_i^{y,0}, v_i^{z,0}) = (\sqrt{T_x^0}\mathcal{N}_i^{x,0}, \sqrt{T_y^0}\mathcal{N}_i^{y,0}, \sqrt{T_z^0}\mathcal{N}_i^{z,0}),$$

where $\mathcal{N}_i^{x,0}, \mathcal{N}_i^{y,0}, \mathcal{N}_i^{y,0}$ are samples of $\mathcal{N}(0, 1)$. For the parameter $m = T_p^0$, we can then compute

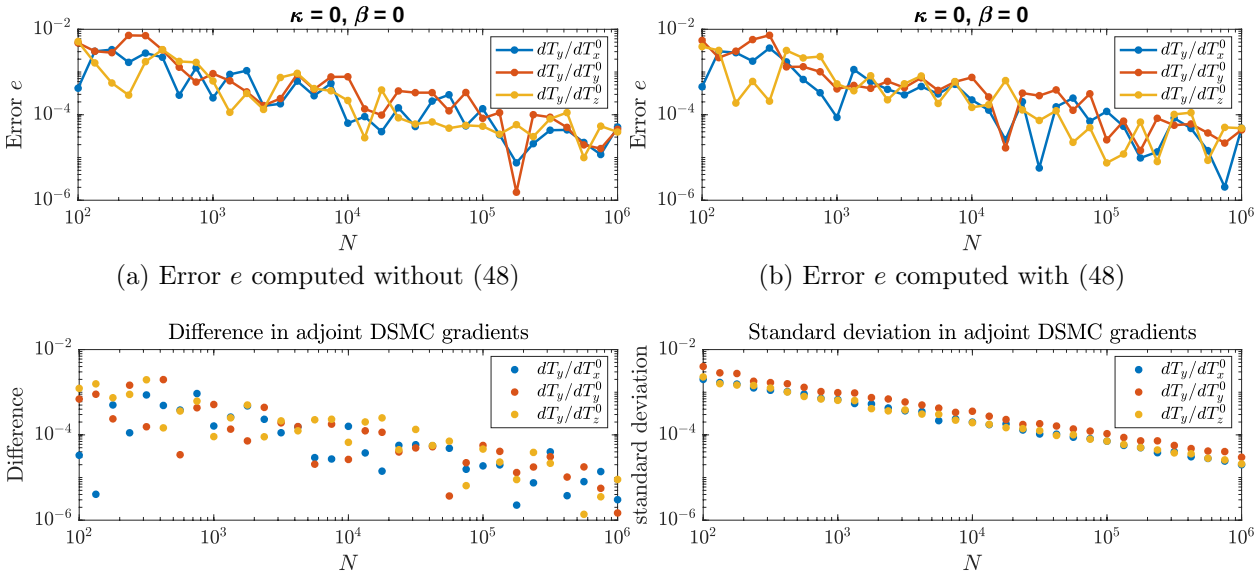
$$\nabla_m v_i^{j,0} = \frac{\mathcal{N}_i^{j,0}}{2\sqrt{T_p^0}}\delta_{j,p} = \frac{v_i^{j,0}}{2T_p^0}\delta_{j,p}, \quad j, p \in \{x, y, z\}.$$

This way of obtaining $\nabla_m v_i^{j,0}$ corresponds to the so-called pathwise gradient estimator [16, Section 5].

4.1. Simulations with Angle-Independent Collision Kernels

First, we focus on the angle-independent collision kernel by setting $\kappa = 0$ and $\epsilon = 10$. Thus, the collision kernel $q(v - v_1, \sigma) = 1/(40\pi)|v - v_1|^\beta$ following (51).

When $\beta = 0$, the collision kernel corresponds to Maxwell molecules discussed in [9]. We focus on the objective function T_y as an example. After $M = 20$ time steps of the forward DSMC simulation using different numbers of particles N, we numerically illustrate the error in the gradient calculation both with and without the term (48) in the adjoint equation; see Figures 1a and 1b. The differences in the averaged adjoint DSMC gradients $\overline{\nabla_m J}^{AD}$, as seen in Figure 1c, are observed to be in the same order as the random errors in $\overline{\nabla_m J}^{AD}$, as shown in Figure 1d. The comparison in Figure 1 illustrates that the additional term does not affect angle-independent kernels due to an averaging effect. This phenomenon occurs because, for angle-independent collision kernels, the post-collision relative velocity σ can be seen as uniformly distributed over the sphere, which weakens its dependence on the scattering angle θ and the pre-collision relative velocity α, despite the relation $\cos\theta = \sigma \cdot \alpha$. In [9], we assumed that σ does not depend on α for the Maxwellian gas and obtained the adjoint DSMC algorithm without the term (48). Here, we use this example to illustrate that both adjoint matrices are valid for angle-independent kernels.



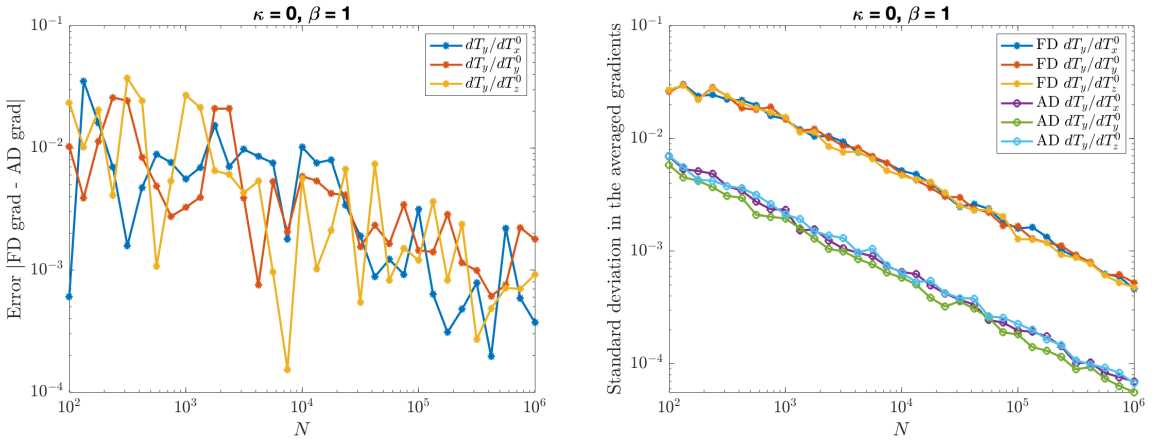
(a) Error e computed without (48)

(b) Error e computed with (48)

(c) Differences in $\overline{\nabla_m J}^{AD}$ with and without (48)

(d) The random errors in $\overline{\nabla_m J}^{AD}$

Figure 1: (a)-(b): Comparison of the gradient error (53) for the Maxwellian constant collision kernel ($\kappa = 0$, $\beta = 0$) with and without the term (48) in the adjoint matrix (47) in the adjoint DSMC Algorithm 2. (c): The absolute value of the difference in $\overline{\nabla_m J}^{AD}$ between the two cases. (d): The random error in $\overline{\nabla_m J}^{AD}$ measured by the estimated standard deviation in the averaged value $\overline{\nabla_m J}^{AD}$ from $M_s = 100$ i.i.d. runs. Note that the term (48) is deterministic and thus does not affect the random error in the adjoint DSMC gradient.



(a) Gradient error e in (53)

(b) Standard deviations of $\overline{\nabla_m J}^{FD}$ and $\overline{\nabla_m J}^{AD}$

Figure 2: The gradient error (a) and the standard deviation (b) for the case $\beta = 1$ and $\kappa = 0$ after 20 time steps. The standard deviations plotted in (b) are for the averaged gradient values from $M_s = 100$ i.i.d. runs.

Next, we consider the case $\beta = 1$. The objective function is T_y, the final temperature in the y direction. In Figure 2a, we show that the gradient error e, defined in (53), decays as the number of particles N increases, while Figure 2b illustrates that the standard deviations of the averaged adjoint DSMC gradient $\overline{\nabla_m J}^{AD}$ and finite-difference gradient $\overline{\nabla_m J}^{FD}$ both decay as $\mathcal{O}(1/\sqrt{N})$. As mentioned earlier, the error e observed in Figure 2a has three contributions: the random error contributions from both $\nabla_m J^{AD}$ and $\nabla_m J^{FD}$, and the finite difference error in $\nabla_m J^{FD}$. We remark that, based on the standard deviations shown in Figure 2b, the random error in $\nabla_m J^{FD}$ could be a leading contribution in e and the random error in the adjoint DSMC gradient $\nabla_m J^{AD}$ is nearly 10 times smaller.

In Figure 3, we consider the case $\beta = 2$, $\kappa = 0$, and evaluate the gradient $\frac{\partial T_l}{\partial T_p^0}$, where $l, p \in \{x, y, z\}$. We fix the number of particles $N = 10^6$, and investigate how the gradient error e defined in (53) and the standard deviations of $\overline{\nabla_m J}^{AD}$ and $\overline{\nabla_m J}^{FD}$ change with respect to the total number of time steps M (the x axis in the plots). We consider $M = 1, \ldots, 20$. In Figure 3a, the gradient errors are mostly linearly increasing up to perturbations incurred by the random errors (illustrated in Figure 3b). Based on the forward DSMC Algorithms 1 and 3, the variations of both gradients, $\overline{\nabla_m J}^{AD}$ and $\overline{\nabla_m J}^{FD}$, are expected to be $\mathcal{O}(M)$ since the number of sampling steps in the forward DSMC algorithms grows linearly in time. Therefore, we observe from the log-log plots in Figure 3b that the standard deviations for both gradients grow as $\mathcal{O}(\sqrt{M})$, while the standard deviation of the finite difference gradient $\overline{\nabla_m J}^{FD}$ is much bigger than the one for the adjoint DSMC gradient $\overline{\nabla_m J}^{AD}$.

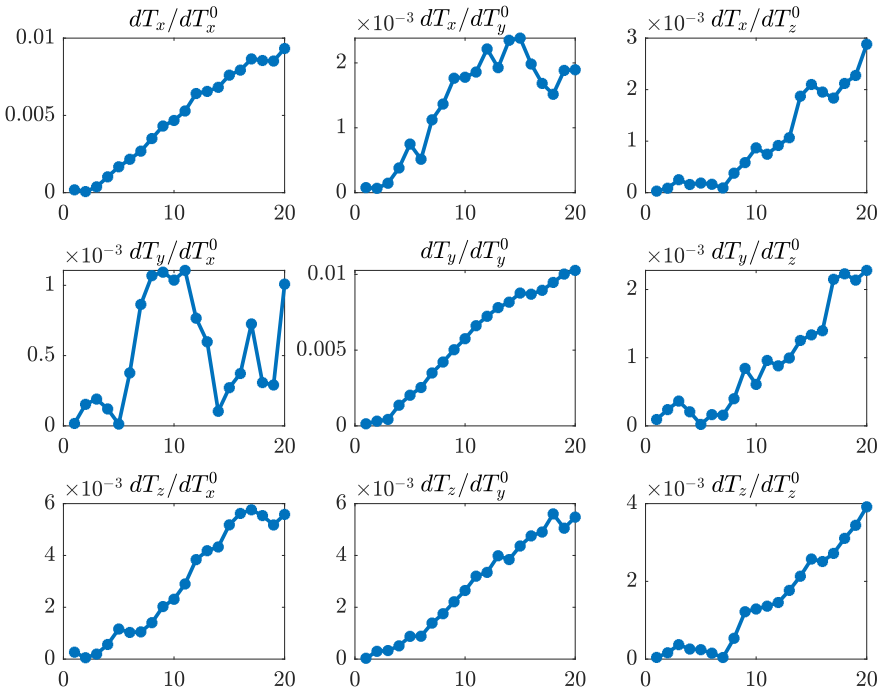
4.2. Simulations with Angle-Dependent Collision Kernels

In this subsection, we consider collision kernels that are both velocity and angle-dependent. We set $\beta = 1$, $\epsilon = 10$ and consider various κ values. The collision kernel takes the form

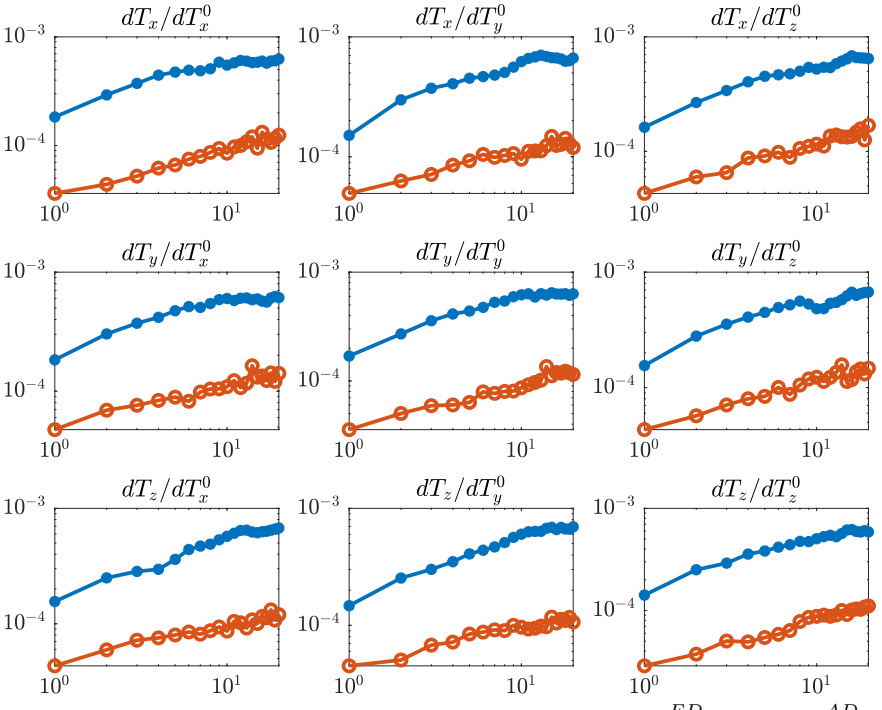
$$q(v - v_1, \sigma) = \frac{(1+\kappa)}{10\pi\, 2^{\kappa+2}}(1 + \cos\theta)^\kappa |v - v_1|, \quad \cos\theta = \sigma \cdot \frac{v - v_1}{|v - v_1|}.$$

We remark that when the collision kernel is angle-dependent, the adjoint matrix D_i^k in Algorithm 2 should follow eq. (47) instead of eq. (45). Note that the difference between the two adjoint matrices is that eq. (47) has an additional term (48).

We first consider cases $\kappa = 1$ and $\kappa = 2$, and set the objective function as T_y with the parameter $m = T_p^0$, where $p \in \{x, y, z\}$. We plot the gradient errors in Figure 4, both of which decay as the number of particles N increases. In Figure 5, we focus on the case $\kappa = 5$ and evaluate the gradient $\frac{\partial T_l}{\partial T_p^0}$ where $l, p \in \{x, y, z\}$, after $M = 20$ time steps. Similar to Figure 1, we compare the gradient errors when the adjoint DSMC gradients $\nabla_m J^{AD}$ are computed with the adjoint matrix (45) and (47), respectively. In Figure 5a, all 9 gradient errors plateaued at a relatively large constant after $N \geq 10^4$. On the other hand, in Figure 5b, all 9 gradient errors asymptotically decay as the number of particles N increases. All the gradient errors are about 1×10^{-5} when $N = 10^6$. The finite difference gradients $\nabla_m J^{FD}$ are the same in both figures, so the drastic differences in e come from the adjoint DSMC gradients $\nabla_m J^{AD}$. The comparison in Figure 5 indicates that the adjoint matrix D defined in (45) is incorrect for angle-dependent kernels, which leads to wrong adjoint DSMC gradients in Figure 5a. The difference between Figure 1 and Figure 5 also shows that the additional term (48) is crucial for angle-dependent kernels, but plays little role for angle-independent collision kernels.

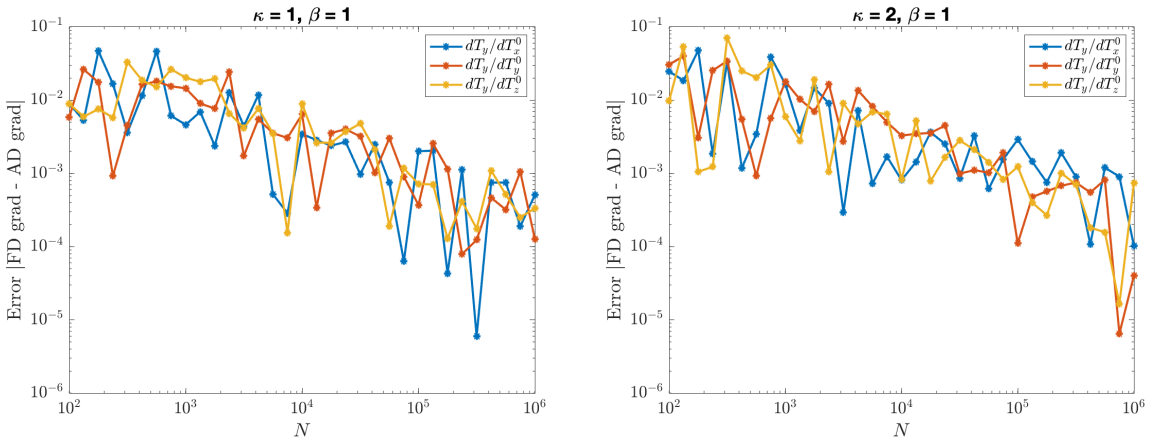


(a) Error e defined in (53)



(b) Standard deviations in the averaged gradients $\overline{\nabla_m J}^{FD}$ and $\overline{\nabla_m J}^{AD}$

Figure 3: The gradient error (a) and the standard deviation (b) of $\frac{\partial T_l}{\partial T_p^0}$, $l, p \in \{x, y, z\}$, for the case $\beta = 2$, $\kappa = 0$ and $N = 10^6$, with the number of time steps M ranging from 1 to 20 (the x axis). In Figure 3b, the blue log-log plots represent the standard deviations of $\overline{\nabla_m J}^{FD}$ while the red log-log plots are for $\overline{\nabla_m J}^{AD}$.



(a) Gradient error e for $\kappa = 1$, $\beta = 1$ (b) Gradient error e for $\kappa = 2$, $\beta = 1$

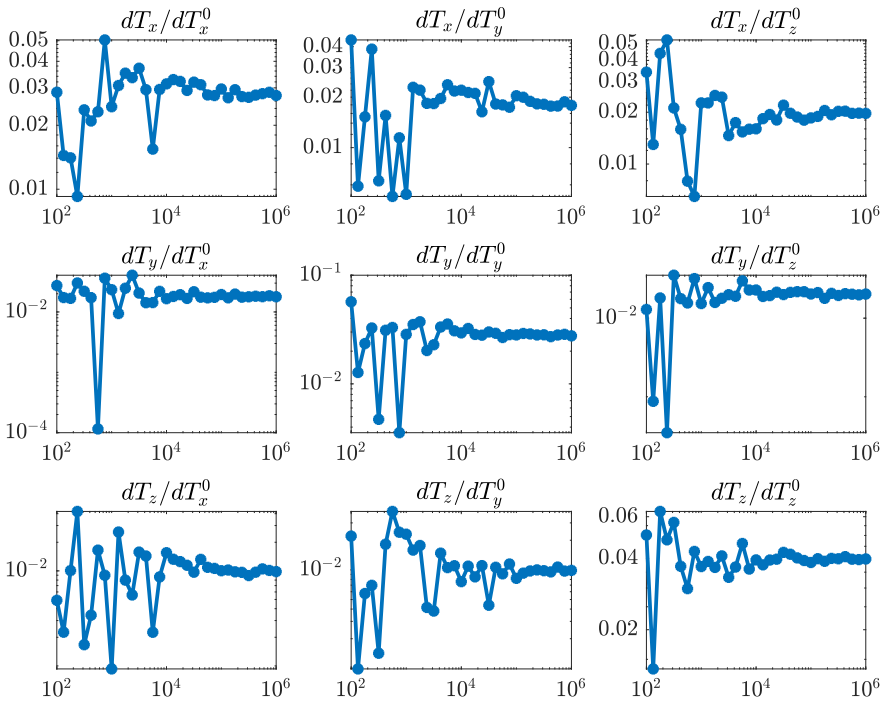
Figure 4: The gradient error (53) (the y axis) for the collision kernel (51) with $\beta = 1$, $\kappa = 1$ (left) and $\kappa = 2$ (right), after $M = 20$ time steps. The number of particles N (the x axis) ranges from 10^2 to 10^6.

4.3. Comments on the General Kernel Case

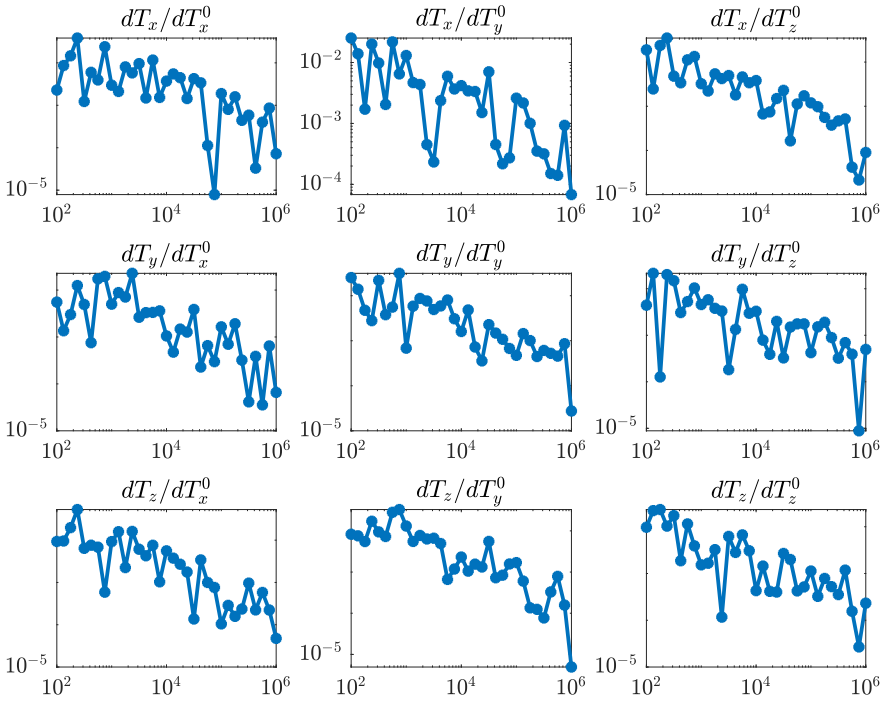
All the tests above were performed for separable collision kernels of type (8) and therefore were based on Algorithm 3 for the forward DSMC simulations and Algorithm 2 for the adjoint DSMC simulations. For general collision kernels $q(v - v_1, \sigma) = \tilde{q}(|v - v_1|, \theta)$ that satisfy (7), we can use Algorithm 1 for the forward DSMC simulations together with Algorithm 2 for the adjoint DSMC simulations. To this end, we have also numerically verified an approach of computing the forward DSMC via Algorithm 1 (and corresponding adjoint DSMC via Algorithm 2) that does not split the kernel (8) into velocity-dependent and angle-dependent parts, but rather performs rejection sampling over the full collision kernel with the scattering angle σ uniformly sampled over the unit sphere (as described in in Algorithm 1). In this case, one must modify the η_i^k term in eq. (37) used in Algorithm 2. Previously, it was based on rejection sampling using $q \sim |v - v_1|^\beta$, whereas in this case it is based on $q \sim |v - v_1|^\beta (1 + \cos\theta)^\kappa$.

Based on our numerical results, we conclude that we still need the extra term (48) in (47) when running Algorithm 2 for an angle-dependent collision kernel. Even though the collision parameters σ for collisions are sampled uniformly in the forward DSMC Algorithm 1 and do not *explicitly* depend on the collision velocity pair (v, v_1), after the rejection sampling step, the unit vector σ becomes *implicitly* dependent on (v, v_1). The only case where the term (48) is not needed in the adjoint matrix D is when the collision kernel is angle-independent; see Figure 1 for an illustration.

It is worth noting that, for kernel q in the form of (8) and (51), Algorithm 3 (which takes advantage of the separable form of q) is more efficient than the forward DSMC Algorithm 1 designed for a more general kernel q, except in the angle-independent case, where the two algorithms are essentially identical. This is because Algorithm 1 involves sampling more virtual collisions and taking additional time to sample collision parameters σ for all virtual collision pairs instead of sampling collision parameters only for the actual collisions (as done in Algorithm 3). For example, when using kernel (51) with $\kappa = 2$, $\beta = 1$, Algorithm 3



(a) Error e computed without (48)



(b) Error e computed with (48)

Figure 5: Comparison of the gradient error (53) (the y axis) for the collision kernel with $\kappa = 5$, $\beta = 1$ after $M = 20$ time steps, with and without the additional term (48) in the adjoint matrix (47) in the adjoint DSMC Algorithm 2 for gradient calculation. The number of particles N (the x axis) ranges from 10^2 to 10^6. The gradient errors are large without (48) in the adjoint matrix, which leads to a wrong adjoint equation.

computes approximately 3 times faster than Algorithm 1, while they both provide error of order 10^{-4} in the adjoint DSMC gradient $\nabla_m J^{AD}$ for $N = 10^6, M = 20, \Delta t = 0.1, M_s = 100$. We will analyze the computational cost and memory requirements for these two variants later in Section 4.4. Nevertheless, Algorithm 1 can be used for general collision kernels (3) as long as the condition (7) is satisfied (which is generally true for numerical computations).

4.4. Discussion on the Computational Cost or Memory Requirements

This subsection discusses the computational cost and memory requirements associated with our forward and adjoint DSMC algorithms.

4.4.1. Memory Requirements

The forward DSMC simulation based on Algorithm 1 is done using N particles. Thus, we need $3N \times 8 = 24N$ bytes in double-precision arithmetic to store the velocities of all the particles. At each time step, $2N_c = \Delta t \mu N$ particles virtually collide, which is usually a small fraction of N. Thus, we disregard the temporary variables used to compute the collisions in calculating the overall memory requirements. We also override the particle velocities each time step to avoid using extra memory for the new velocities.

The adjoint DSMC using Algorithm 2 requires $3N \times 8 = 24N$ bytes for the storage of $\gamma_i^k \in \mathbb{R}^3$. They could be stored in the same memory location for the final particle velocities v_i^M as soon as η_i^k are also computed at the final time $t = t_M$. To later compute the gradient (41) using the adjoint DSMC algorithm, we also need to store $\frac{\partial \hat{v}_{0i}(m)}{\partial m}$ ($24N$ bytes if m is a scalar) and the following additional information for each colliding pair at each time step during the forward DSMC simulation:

- the integer-valued indices $\{i, i_1\}$ of the colliding particles, 4 bytes for each if they are stored in the UInt32 format that allows values up to $2^{32} - 1 \approx 4.3 \times 10^9$;
- the σ vector in $\mathcal{S}^2$ represented by the polar coordinate, 8 bytes for each coordinate;
- the unit collision direction $\alpha = \frac{v - v_1}{|v - v_1|} \in \mathcal{S}^2$ represented by the polar coordinate, 8 bytes for each coordinate;
- $\eta_i^k, \eta_{i_1}^k \in \mathbb{R}^3$ as defined in (37), 8 bytes for each vector component.

In total, the memory requirements are 88 bytes per colliding pair or 44 bytes per colliding particle. Overall, we need $88N_c$ new bytes stored per time step. If the number of time steps is $M = T/\Delta t$, the total amount of extra storage needed for the backward propagation is $88N_cM = 44N\mu\Delta tM = 44N\mu T$. It becomes comparable to the $24N$ bytes needed for the particle storage in the forward DSMC when we have $\mu T = \mathcal{O}(1)$. Note that Algorithm 1 and Algorithm 3 have two different μ-values. The total amount of extra memory required for the backward-in-time adjoint propagation is $24N + 44N\mu T$ bytes, and the total amount of memory required for both the forward DSMC and the adjoint DSMC simulations is $48N + 44N\mu T$ bytes. To compute the gradient, we require another $24Nd_m$ bytes of memory where d_m is the dimension of the parameter m.

4.4.2. Computational Cost

In this subsection, we will discuss the computational cost in computing the gradient through Algorithm 2 using both Algorithm 1 and Algorithm 3 as the forward DSMC algorithm. We will use superscripts "A1" and "A3" in symbols such as μ, N_c and ν to indicate that they are different quantities from the two different algorithms.

For for a separable collision kernel of type (51), when using Algorithm 3 for the forward DSMC simulation and Algorithm 2 for the adjoint DSMC, we do

- 12 operations and 3 random numbers (2 indices and 1 for rejection sampling) per virtual collision pair to select actual collision pairs;
- 59 operations and 2 random numbers (angles θ_i^k and φ_i^k) per collision pair to collide two particles;
- 15 operations per virtual collision pair to compute pre-factors $\frac{\partial \log q_i^k}{\partial v_i^k}, \frac{\partial \log(\Sigma - q_i^k)}{\partial v_i^k}$ for $\eta_i^k, \eta_{i_1}^k$ in (37);
- 92 operations to collide two γ-particles during the adjoint DSMC algorithm step if the actual collision occurred between the corresponding v-particles in the forward DSMC process, and 6 operations if it did not.

Every time step, we have in total $(12+59\nu^{A3}+15+92\nu^{A3}+6(1-\nu^{A3}))N_c^{A3} = \left(33+145\nu^{A3}\right) N_c^{A3}$ flops (floating point operations) and generate $(3+2\nu^{A3})N_c^{A3}$ random numbers, where $N_c^{A3} = \Delta t \mu^{A3} N/2$ and ν^{A3} represents a fraction of particles that participated in the actual collisions out of all virtual pairs. Note that ν^{A3} depends on the collision kernel, its associated parameters and the current velocity distribution (namely, Σ_v, β and the distribution of $\mathcal{V}^k$). Based on the definition of μ^{A3} in (10) and the particular form of the collision kernel (51), we know that $\mu^{A3} = \Sigma_v/\epsilon = \max|v - v_1|^\beta/\epsilon \approx 10^\beta/\epsilon$ (in our simulations we assume that $|v - v_1| \approx 10$ since the distribution $f(v,t)$ is close to Maxwellian with width 1 for all times t). If we further assume that ν^{A3} and μ^{A3} remain approximately the same among different time steps, for a total of $M = T/\Delta t$ time steps, we have $\left(33+145\nu^{A3}\right)\mu^{A3}NT/2$ flops, and $(3+2\nu^{A3})\mu^{A3}NT/2$ random number generations. We count the random number generations separately since these operations take much more CPU time and are approximately 10-100 flops each, based on our observations from running the numerical tests.

On the other hand, for a kernel of type (51), if we use Algorithm 1 for the forward DSMC simulation and Algorithm 2 for the adjoint DSMC, we do

- 15 operations and 4 random numbers (2 indices, 1 for θ_i^k, 1 for rejection sampling) per virtual collision pair to select actual collision pairs;
- 60 operations and 1 random number (angle φ_i^k) per collision pair to collide two particles;
- 17 operations per virtual collision pair to compute pre-factors $\frac{\partial \log q_i^k}{\partial v_i^k}, \frac{\partial \log(\Sigma - q_i^k)}{\partial v_i^k}$ for $\eta_i^k, \eta_{i_1}^k$ in (37);

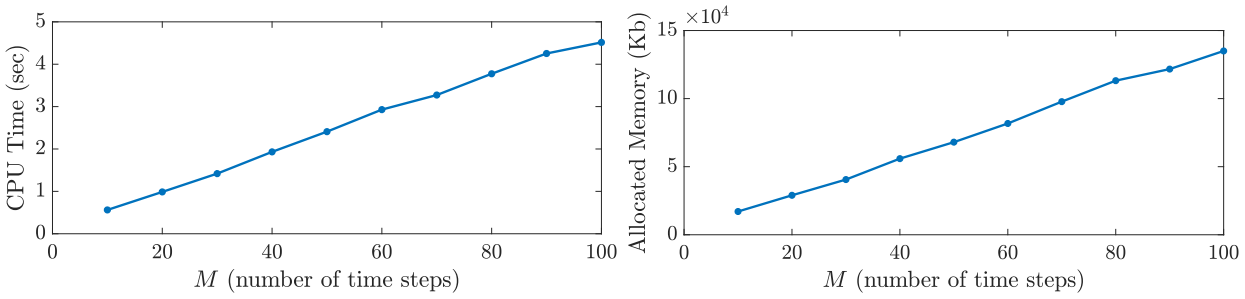


Figure 6: Illustration of the computation cost and memory requirement for a single evaluation of the adjoint DSMC gradient with respect to the number of time steps (ranging from 10 to 100), $\Delta t = 0.1$. Here, we use $N = 10^4$ particles and set $T_0 = (1, 1, 0.5)$, $\kappa = 0$, $\beta = 2$ and $\epsilon = 10$. We remark that both the CPU time and the memory requirements are $\mathcal{O}(T)$ where $T = M\Delta t$ is the total simulation time.

- 92 operations to collide two γ-particles during the adjoint DSMC algorithm step if the actual collision occurred between the corresponding v-particles in the forward DSMC process, and 6 operations if it did not.

Thus, we have in total $(15{+}60\nu^{A1}{+}17{+}92\nu^{A1}{+}6(1{-}\nu^{A1}))N_c^{A1} = (38{+}146\nu^{A1})N_c^{A1}$ operations and $(4+\nu^{A1})N_c^{A1}$ random number generations, where $N_c^{A1} = \Delta t \mu^{A1} N/2$ and ν^{A1} is roughly a fraction of particles that participated in the actual collisions out of all virtual pairs. Notice that here $\mu^{A1} = 4\pi\Sigma = 4\pi \max|q(v - v_1, \sigma)| = (1+\kappa)\max|v - v_1|^\beta/\epsilon \approx (1+\kappa)10^\beta/\epsilon$. Again, if we assume that ν^{A1} and μ^{A1} stay roughly the same among all $M = T/\Delta t$ time steps, we have in total $(38 + 146\nu^{A1})\mu^{A1}NT/2$ flops and $(4+\nu^{A1})\mu^{A1}NT/2$ random number generations in the forward and adjoint DSMC simulations.

Next, we compare the gradient computations when using Algorithm 1 and Algorithm 3 as the forward DSMC algorithm. First, we point out that $\mu^{A1} \geq \mu^{A3}$, as the former is about $(1+\kappa)10^\beta/\epsilon$ while the latter is about $10^\beta/\epsilon$. On the other hand, the maximum value of the sampling distribution used in the rejection sampling for Algorithm 1 is $\Sigma \approx (1+\kappa)10^\beta$, while the value is $\Sigma_v \approx 10^\beta$ for the rejection sampling in Algorithm 3. As a result, the accepted particles are much fewer in Algorithm 1 compared to those in Algorithm 3, i.e., $\nu^{A1} \leq \nu^{A3}$. We observe that overall $\mu^{A1}\nu^{A1} \approx \mu^{A3}\nu^{A3}$. That is, the two algorithms share a similar number of actual collisions while Algorithm 1 incurs $(1+\kappa)$ times more virtual collisions than Algorithm 3. Consequently, Algorithm 1 incurs more flops and random number generations than Algorithm 3 for the same separable collision kernel (51) when $\kappa > 0$. Recall that when $\kappa = 0$, these two algorithms become the same.

Compared to [9], which studied the adjoint DSMC algorithm for Maxwellian particles, we use about two times more memory, operations, and random number generations to deal with the more general collision kernel in both the forward and the adjoint DSMC (using Algorithm 3 and Algorithm 2, respectively). However, for the special case of $\kappa = \beta = 0$, i.e., the Maxwellian particles, one can adapt the current framework by switching to the more efficient algorithm in [9] through an if-then statement in the algorithm. This way, we can still take advantage of the extra efficiency when handling Maxwellian particle collisions.

4.4.3. Numerical Illustration

In Figure 6, we show the CPU time and memory requirements for a single run to compute the adjoint DSMC gradient for a various number of time steps M by fixing the number

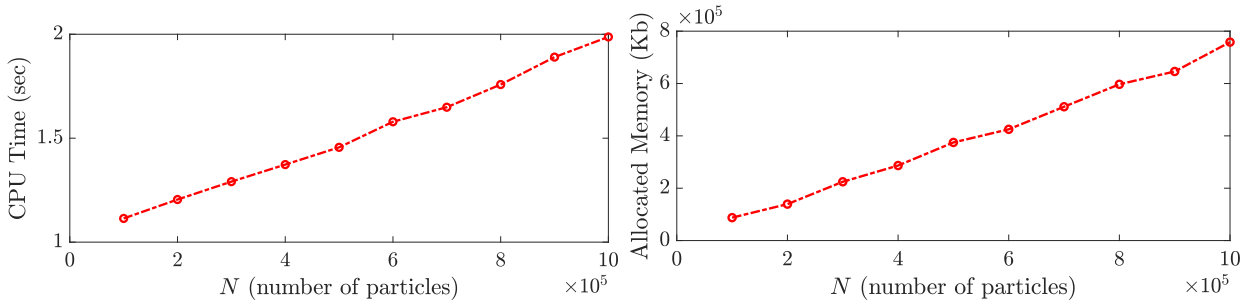


Figure 7: Illustration of the computation cost and memory requirement for a single evaluation of the adjoint DSMC gradient with respect to the number of particles (ranging from 10^5 to 10^6). Here, we use $M = 20$ time steps $\Delta t = 0.1$ and set $T_0 = (1, 1, 0.5)$, $\kappa = 0$, $\beta = 2$ and $\epsilon = 10$.

of particles N, while in Figure 7, we fix M and vary N. In these simulations we used $\kappa = 0, \beta = 2, \epsilon = 10$, so $\mu = \mu_\kappa = \Sigma_v/\epsilon \approx 10^\beta/\epsilon = 10$. The total amount of memory required for the adjoint DSMC was about 40 times the amount required for the forward DSMC when $T = 2$, i.e., $M = 20$. As we can observe from these plots, both the computational cost and the memory requirements in the adjoint DSMC algorithm grow linearly with respect to the number of time steps M and the number of particles N. However, we want to point out that, based on our analysis, despite their linear growth with respect to M, the computational cost and memory requirement is $\mathcal{O}(T)$ where $T = M\Delta t$ is total simulation time.

5. Conclusions

As discussed in Section 1 and Section 3.1.4, the method developed in this work is mainly based on the DTO approach but also involves an OTD step. The reason for the OTD step is that the rejection sampling involves a decision of whether a virtual collision is real, and the decision depends on the unknown parameter for which we want to compute the gradient. Differentiation after this choice would be applied to a discontinuous function (to indicate whether a collision is real or virtual) of the particle velocity, leading to a singularity. The OTD step in the new adjoint DSMC method, i.e., first differentiating the expectation over the decision and then sampling the resulting gradient, enables us to circumvent the difficulty of directly differentiating a discontinuous decision function from the rejection sampling.

Another main contribution of this work is to consider collision kernels that are also scattering angle-dependent. As a result, the post-collision relative velocity depends on the pre-collision relative velocity. Although in the angle-independent cases, such as a constant Maxwellian collision kernel, the post-collision relative velocity always depends on the scattering angle as well as the pre-collision velocities based on their definitions, we can ignore this dependence due to the averaging effect since the post-collision relative velocity is uniformly distributed over the sphere. This is not the case for angle-dependent kernels. In our new derivations for the adjoint DSMC algorithm, the resulting adjoint equation has an additional term that reflects this dependence. We remark that this additional term can be efficiently computed without extra memory requirement.

This paper extends the adjoint DSMC method to a much more general class of collision kernels for the Boltzmann equation than the original proposal [9]. In future works, we plan

to extend the adjoint DSMC method, for example, to Coulomb collisions and apply the method to large-scale optimization problems constrained by Boltzmann equations.

Acknowledgements

This material is based upon work supported by the National Science Foundation under Award Number DMS-1913129 and the U.S. Department of Energy under Award Number DE-FG02-86ER53223. Y. Yang acknowledges support from Dr. Max Rössler, the Walter Haefner Foundation and the ETH Zürich Foundation.

References

[1] Albi, G., Bellomo, N., Fermo, L., Ha, S.Y., Kim, J., Pareschi, L., Poyato, D., Soler, J., 2019. Vehicular traffic, crowds, and swarms: From kinetic theory and multiscale methods to applications and research perspectives. Mathematical Models and Methods in Applied Sciences 29, 1901–2005.

[2] Albi, G., Herty, M., Pareschi, L., 2015. Kinetic description of optimal control problems and applications to opinion consensus. Communications in Mathematical Sciences 13, 1407–1429.

[3] Albi, G., Pareschi, L., Zanella, M., 2014. Boltzmann-type control of opinion consensus through leaders. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences 372, 20140138.

[4] Babovsky, H., Illner, R., 1989. A convergence proof for Nanbu's simulation method for the full Boltzmann equation. SIAM Journal on Numerical Analysis 26, 45–65.

[5] Babovsky, H., Neunzert, H., 1986. On a simulation scheme for the Boltzmann equation. Mathematical Methods in the Applied Sciences 8, 223–233.

[6] Ben Abdallah, N., Degond, P., Génieys, S., 1996. An energy-transport model for semiconductors derived from the Boltzmann equation. Journal of Statistical Physics 84, 205–231.

[7] Bird, G., 1970. Direct simulation and the Boltzmann equation. The Physics of Fluids 13, 2676–2681.

[8] Burini, D., De Lillo, S., Gibelli, L., 2016. Collective learning modeling based on the kinetic theory of active particles. Physics of Life Reviews 16, 123–139.

[9] Caflisch, R., Silantyev, D., Yang, Y., 2021. Adjoint DSMC for nonlinear Boltzmann equation constrained optimization. Journal of Computational Physics 439, 110404.

[10] Cordier, S., Pareschi, L., Toscani, G., 2005. On a kinetic model for a simple market economy. Journal of Statistical Physics 120, 253–277.

[11] Davis, A.D., Giannakis, D., Stadler, G., Stechmann, S.N., Manucharyan, G., 2020. Super-parameterization of Lagrangian sea ice dynamics using the Boltzmann equation, in: AGU Fall Meeting Abstracts, pp. C048–03.

[12] Gamba, I.M., Haack, J.R., Hauck, C.D., Hu, J., 2017. A fast spectral method for the Boltzmann collision operator with general collision kernels. SIAM Journal on Scientific Computing 39, B658–B674.

[13] Guan, K., Matsushima, K., Noguchi, Y., Yamada, T., 2023. Topology optimization for rarefied gas flow problems using density method and adjoint IP-DSMC. Journal of Computational Physics 474, 111788.

[14] Kanazawa, K., Sueshige, T., Takayasu, H., Takayasu, M., 2018. Derivation of the Boltzmann equation for financial Brownian motion: Direct observation of the collective motion of high-frequency traders. Physical Review Letters 120, 138301.

[15] Koura, K., Matsumoto, H., 1991. Variable soft sphere molecular model for inverse-power-law or lennard-jones potential. Physics of fluids A: fluid dynamics 3, 2459–2465.

[16] Mohamed, S., Rosca, M., Figurnov, M., Mnih, A., 2020. Monte Carlo gradient estimation in machine learning. Journal of Machine Learning Research 21, 1–62.

[17] Naesseth, C., Ruiz, F., Linderman, S., Blei, D., 2017. Reparameterization gradients through acceptance-rejection sampling algorithms. Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS) .

[18] Nanbu, K., 1980. Direct simulation scheme derived from the Boltzmann equation. I. monocomponent gases. Journal of the Physical Society of Japan 49, 2042–2049.

[19] Pareschi, L., Russo, G., 2001. An introduction to Monte Carlo method for the Boltzmann equation, in: ESAIM: Proceedings, EDP Sciences. pp. 35–75.

[20] Pareschi, L., Toscani, G., 2013. Interacting multiagent systems: kinetic equations and Monte Carlo methods. OUP Oxford.

[21] Pareschi, L., Toscani, G., 2014. Wealth distribution and collective knowledge: a Boltzmann approach. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences 372, 20130396.

[22] Poupaud, F., 1991. Diffusion approximation of the linear semiconductor Boltzmann equation: analysis of boundary layers. Asymptotic Analysis 4, 293–317.

[23] Villani, C., 2002. A review of mathematical topics in collisional kinetic theory, in: Friedlander, S., Serre, D. (Eds.), Handbook of Mathematical Fluid Dynamics. volume 1, pp. 71–305.

[24] Wang, C., Lin, T., Caflisch, R., Cohen, B.I., Dimits, A.M., 2008. Particle simulation of Coulomb collisions: Comparing the methods of Takizuka & Abe and Nanbu. Journal of Computational Physics 227, 4308–4329.