Journal of Computational Physics 439 (2021) 110404

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

Adjoint DSMC for nonlinear Boltzmann equation constrained optimization

Russel Caflisch, Denis Silantyev, Yunan Yang *

Courant Institute of Mathematical Sciences, New York University, New York, NY 10012, United States of America

ARTICLE INFO

Article history:
Available online 4 May 2021

Keywords:
Boltzmann equation
Direct simulation Monte Carlo methods
DSMC
Optimization
Adjoint-state method
Linear Boltzmann equation

ABSTRACT

Applications for kinetic equations such as optimal design and inverse problems often involve finding unknown parameters through gradient-based optimization algorithms. Based on the adjoint-state method, we derive two different frameworks for approximating the gradient of an objective functional constrained by the nonlinear Boltzmann equation. While the forward problem can be solved by the DSMC method, it is difficult to efficiently solve the high-dimensional continuous adjoint equation obtained by the "optimize-then-discretize" approach. This challenge motivates us to propose an adjoint DSMC method following the "discretize-then-optimize" approach for Boltzmann-constrained optimization. We also analyze the properties of the two frameworks and their connections. Several numerical examples are presented to demonstrate their accuracy and efficiency.

1. Introduction

The development of modern technology requires accuracy in modeling physical processes. One critical task is to model the kinetic behavior of rarefied gas, which is required for low-pressure gas flow and cannot be accurately described by the Navier–Stokes equations. The Boltzmann equation models the dynamics of a many-particle system through a velocity distribution function, with a nonlinear collision operator that describes binary particle interactions. The Boltzmann equation is a powerful tool from the kinetic theory describing molecular gas dynamics, radiative transfer, plasma physics, and grain and polymer flow [4].

The Boltzmann equation can also be used for design, optimization, control, and inverse problems. A few of the many examples include the design of the semiconductor device, the topology optimization of the gas flow channel, and risk management in quantitative finance [22,23,21,42,31]. One may use a kinetic perspective to tackle optimal control problems for a large system of interacting agents. Common kinetic models involve the Boltzmann model and some mean-field models [44,17,3,24,25,2]. Many of these applications involve finding unknown or optimal parameters in Boltzmann-type equations such that an objective function formed by the computational or experimental data is optimized, i.e., optimization problems with PDE constraints. However, due to the challenges caused by the complex nonlinear collision term, the linear Boltzmann equation [22] or simplified collision operators such as the Bhatnagar–Gross–Krook (BGK) model [8,42], the Ellipsoidal Statistical model [30], or the Shakhov model [43] have been used as alternatives. Although these simplified models provide good approximations for many scenarios, the original nonlinear collision operator is still preferred for better accuracy [19].

* Corresponding author.
E-mail addresses: caflisch@courant.nyu.edu (R. Caflisch), silantyev@courant.nyu.edu (D. Silantyev), yunan.yang@nyu.edu (Y. Yang).

https://doi.org/10.1016/j.jcp.2021.110404
0021-9991/

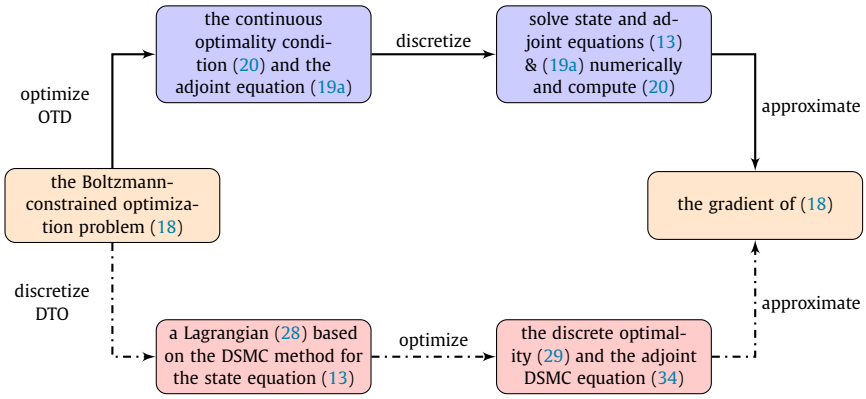

Fig. 1. A diagram summarizing the OTD approach (solid line) and the DTO approach (dash line) to compute the gradient with respect to the unknown parameter for the optimization problem (18) constrained by the Boltzmann equation (13).

The main contribution of this work involves the derivation of two optimization frameworks based on the spatially homogeneous Boltzmann equation with the nonlinear collision operator and the design of an efficient adjoint DSMC method for the gradient calculation. In this paper, we focus on the collision kernel for Maxwell molecules, but we expect our method to be generalized to other collision kernels. The frameworks should be broadly applicable to optimal control, optimal design, and general computational inverse problems. We employ two different approaches to derive numerical algorithms: optimize-then-discretize (OTD) and discretize-then-optimize (DTO) [29]. Fig. 1 summarizes these two approaches. In the OTD approach, we obtain a continuous adjoint equation as an optimality condition, which we discretize by a grid-based method or a Monte Carlo type method. In the DTO approach, we discretize the Boltzmann equation and the objective function using the direct simulation Monte Carlo method (DSMC) [10,38,15], from which a discrete optimality condition and the adjoint DSMC system are derived. The adjoint DSMC from the DTO approach is radically more cost-effective than the continuous adjoint equation in the OTD approach. To our knowledge, this is the first efficient numerical scheme to compute the gradient for nonlinear Boltzmann equation constrained optimization. In addition to the derivations, we also analyze the properties of the two adjoint systems and investigate connections between them. The analysis provides a better understanding of the adjoint-state method for constrained optimization problems.

The paper is arranged as follows. In Section 2, we briefly review the homogeneous Boltzmann equation with a nonlinear collision operator, the linearization of the collision operator, and the classical DSMC algorithm for the numerical solution. We derive the continuous adjoint equation for the continuous Boltzmann equation in Section 3 following the Lagrangian method. A particle method is then proposed to solve the continuous adjoint equation numerically. In Section 4, we regard the DSMC discretization of the Boltzmann equation as the forward problem and the particle velocities as the state variables. We then obtain equations for the adjoint particles following the adjoint-state method. Since one can compute the gradient of the optimization problem by both the continuous and the discrete adjoint systems, we analyze their properties and prove the critical connections between the two adjoint systems in Section 5. Finally, we show several numerical examples in Section 6 to demonstrate the accuracy of the gradient formulae obtained by both the continuous and the DSMC adjoint systems. Detailed comparisons in terms of memory, accuracy, and computational cost of different numerical schemes are also presented in Section 6. Conclusion and future research directions follow in Section 7. Two appendices provide additional details on the numerical methods and numerical analysis of the results.

2. Boltzmann equation and the direct simulation Monte Carlo (DSMC) method

In this section, we first give a short introduction to the Boltzmann equation and some of its relevant properties. The later part of the section is devoted to the description of the classical DSMC method.

2.1. Boltzmann equation

We consider the Boltzmann equation

$$\frac{\partial f}{\partial t} + v \cdot \nabla_x f = Q(f, f)$$

with the initial condition

$$f(x, v, t = 0) = f_0(x, v),$$

where $f(x,v,t)$ is a nonnegative probability density function that describes the time evolution of the distribution of particles which move with velocity $v \in \mathbb{R}^3$ at the position $x \in \mathbb{R}^3$ at time $t > 0$. The bilinear (nonlinear) collision operator $Q(f,f)$ that describes the binary collisions among particles is defined as follows:

$$Q(f,f) = \int_{\mathbb{R}^3}\int_{\mathcal{S}^2} q(v-v_1,\sigma)(f(v_1')f(v') - f(v_1)f(v))d\sigma dv_1, \tag{1}$$

in which (v', v_1') represent the post-collisional velocities associated with the pre-collisional velocities (v, v_1) and the σ integral is over the surface of unit sphere $\mathcal{S}^2$. By conserving the momentum $v + v_1$ and the energy $v^2 + v_1^2$, we have

$$v' = 1/2(v+v_1) + 1/2|v-v_1|\sigma, \tag{2a}$$

$$v_1' = 1/2(v+v_1) - 1/2|v-v_1|\sigma, \tag{2b}$$

where σ is a collision parameter representing a unit direction of the relative velocity of particles after collision (2). We will hereafter use the shorthand notation f, f_1, f', f_1' to denote $f(v), f(v_1), f(v')$ and $f(v_1')$.

The collision operator Q has the following symmetries, which are related to the conservation of mass, momentum and energy: The transformations

$$v \leftrightarrow v_1 \text{ and } v' \leftrightarrow v_1'$$

$$v \leftrightarrow v' \text{ and } v_1 \leftrightarrow v_1'$$

with v' and v_1' defined by (2) are isometries for $(\sigma, v, v_1) \in \mathcal{S}^2 \times \mathbb{R}^3 \times \mathbb{R}^3$ [18,46]. We define the unit vector along the direction of $v - v_1$ as $\alpha = \frac{v-v_1}{|v-v_1|}$. The scattering angle $\theta = \cos^{-1}(\sigma \cdot \alpha)$. Also, $|v-v_1| = |v'-v_1'|$, and thus we have

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

where $\tilde{q}$ is related to q by a change of variable [46]. It follows that for any measurable function $F(v, v_1, v', v_1')$ under suitable integrability conditions

$$\iiint F(v,v_1,v',v_1')qd\sigma dvdv_1 = \iiint F(v_1,v,v_1',v')qd\sigma dvdv_1 \tag{3a}$$

$$= \iiint F(v',v_1',v,v_1)qd\sigma dvdv_1 \tag{3b}$$

$$= \iiint F(v_1',v',v_1,v)qd\sigma dvdv_1. \tag{3c}$$

These symmetries are used repeatedly through the paper.

The kernel q is a nonnegative function that characterizes the details of the binary interactions. For the Variable Hard Sphere (VHS) model, the collision kernel is

$$q(v-v_1,\sigma) = \tilde{q}(|v-v_1|,\theta) = C_\beta(\theta)|v-v_1|^\beta. \tag{4}$$

In particular, when $\beta = 0$, the collision kernel corresponds to the Maxwellian gas, which (along with C_β being constant) is the model that we focus on in this paper. For the Coulomb interaction, the collision kernel is given by the Rutherford formula

$$q(v-v_1,\sigma) = \tilde{q}(|v-v_1|,\theta) = \frac{1}{|v-v_1|^3 \sin^4(\theta/2)}.$$

2.2. Operator formulation for collisions

In this section, we rewrite the collision rules (2), in an operator formulation that clarifies some properties that will be useful for the analysis in later sections. First we denote $\alpha = (v-v_1)\hat{}$ and $\sigma = (v'-v_1')\hat{}$ using the notation $x\hat{} = x/|x|$, in which the equation for σ and the relation $|v-v_1| = |v'-v_1'|$ follow from (2). As described in [46, P.53], the change of variables

$$(v, v_1, \sigma) \longrightarrow (v', v_1', \alpha) \tag{5}$$

is an involution with unit Jacobian.

We can rewrite (2), in terms of operators, as

$$\begin{pmatrix} v' \\ v_1' \end{pmatrix} = A(\sigma,\alpha)\begin{pmatrix} v \\ v_1 \end{pmatrix}, \quad \begin{pmatrix} v \\ v_1 \end{pmatrix} = B(\sigma,\alpha)\begin{pmatrix} v' \\ v_1' \end{pmatrix}, \tag{6}$$

where

$$A(\sigma,\alpha)=\frac{1}{2}\begin{pmatrix} I+\sigma\alpha^T & I-\sigma\alpha^T \\ I-\sigma\alpha^T & I+\sigma\alpha^T \end{pmatrix},\ B(\sigma,\alpha)=\frac{1}{2}\begin{pmatrix} I+\alpha\sigma^T & I-\alpha\sigma^T \\ I-\alpha\sigma^T & I+\alpha\sigma^T \end{pmatrix}, \tag{7}$$

where I is the identity matrix in $\mathbb{R}^3$ and $B=A^T=A^{-1}$.

Now consider small perturbations δv and δv_1 in the pre-collision velocities v and v_1. Since the collision parameter vector σ is chosen independently of v and v_1 for a Maxwellian gas, there is no need to perturb σ; i.e., $\delta\sigma=0$. Also note that $\alpha=\partial_v|v-v_1|=-\partial_{v_1}|v-v_1|$. The resulting first-order variations $\delta v'$ and $\delta v_1'$ in the post-collision velocities are

$$\begin{pmatrix}\delta v' \\ \delta v_1'\end{pmatrix}=A(\sigma,\alpha)\begin{pmatrix}\delta v \\ \delta v_1\end{pmatrix}. \tag{8}$$

2.3. The linearized collision operator

The linearized collision operator is defined through perturbation theory [18], most frequently by linearization around a Maxwellian equilibrium. In this subsection, we define the linearized operator, due to multiplicative perturbation [5,14] around a non-equilibrium distribution and its adjoint operator under a weighted L^2 inner product.

For a general velocity distribution f, consider the multiplicative perturbation $\tilde{f}=f(1+\psi)$. Since the collision operator $Q(\tilde{f},\tilde{f})$ defined in (1) is bilinear,

$$\begin{aligned} Q(\tilde{f},\tilde{f})-Q(f,f)&=Q(f,f\psi)+Q(f\psi,f)+Q(f\psi,f\psi)\\ &\approx Q(f,f\psi)+Q(f\psi,f), \end{aligned}$$

in which $\approx$ means that quadratic terms in ψ are neglected. We define the linearized operator $L[f]$ based on f and applied to ψ as

$$L[f]\psi=f^{-1}[Q(f,f\psi)+Q(f\psi,f)]. \tag{9}$$

In an original paper of Maxwell in 1866 [35], the Boltzmann equation was written in the weak formulation [46]. Following this approach for a test function $\gamma(v,t)$ and using the symmetries (3), one gets the identity

$$\int_{\mathbb{R}^3} Q(f,f)\gamma(v)dv=\frac{1}{2}\int_{\mathbb{R}^3}\int_{\mathbb{R}^3}\int_{\mathcal{S}^2} ff_1(\gamma'+\gamma_1'-\gamma-\gamma_1)qd\sigma dv_1dv \tag{10}$$

where γ_1,γ' and γ_1' are shorthand notations for $\gamma(v_1,t)$, $\gamma(v',t)$ and $\gamma(v_1',t)$. We will use (10) to find the adjoint of $L[f]$. We replace f by $f(1+\psi)$ in (10) and calculate the first-order terms with respect to ψ. As in the derivation of (9), the first-order terms in the left-hand side of (10) are derived as

$$\int_{\mathbb{R}^3}[Q(f+f\psi,f+f\psi)-Q(f,f)]\gamma(v)dv\approx\int_{\mathbb{R}^3} f(L[f]\psi)\gamma(v)dv.$$

Similarly, the first-order terms in the right-hand side of (10) are derived as

$$\frac{1}{2}\iiint[(f+f\psi)(f_1+f_1\psi_1)-ff_1](\gamma'+\gamma_1'-\gamma-\gamma_1)qd\sigma dv_1dv\approx\iiint ff_1\psi(\gamma'+\gamma_1'-\gamma-\gamma_1)qd\sigma dv_1dv.$$

Combining both sides, we obtain the first-order variation equation

$$\int_{\mathbb{R}^3} f\left((L[f]\psi)\gamma-\psi\int_{\mathbb{R}^3}\int_{\mathcal{S}^2} f_1(\gamma'+\gamma_1'-\gamma-\gamma_1)qd\sigma dv_1\right)dv=0,$$

which is equivalent to the following

$$(L[f]\psi,\gamma)_{L^2(\mu)}=(\psi,L^*[f]\gamma)_{L^2(\mu)},$$

in which

$$L^*[f]\gamma=\int_{\mathbb{R}^3}\int_{\mathcal{S}^2} f_1(\gamma'+\gamma_1'-\gamma-\gamma_1)qd\sigma dv_1, \tag{11}$$

and $(h_1,h_2)_{L^2(\mu)}=\int h_1h_2fdv$ is the inner product for the weighted Hilbert space $L^2(\mu)$ in which $fdv=d\mu$. This shows that operator $L^*[f]$ is the adjoint of $L[f]$.

In the special case that $f = \mathcal{M}$ is a Maxwellian equilibrium distribution, using the symmetries (3) and the equilibrium property $\mathcal{M}\mathcal{M}_1 = \mathcal{M}'\mathcal{M}'_1$,

$$L[\mathcal{M}]\psi = \int_{\mathbb{R}^3}\int_{\mathcal{S}^2} (\psi' + \psi'_1 - \psi - \psi_1)\mathcal{M}(v_1)q(v - v_1, \sigma)d\sigma dv_1. \tag{12}$$

Comparison of (12) and (11) shows that $L^*[\mathcal{M}] = L[\mathcal{M}]$; i.e., that $L[\mathcal{M}]$ is self adjoint. The linearized collision operator around a Maxwellian has been extensively studied in the literature [18,46,11].

2.4. The direct simulation Monte Carlo (DSMC) method

In this section, we describe the classical DSMC method [10] for the spatially homogeneous Boltzmann equation following the presentation of [40]:

$$\frac{\partial f}{\partial t} = Q(f, f). \tag{13}$$

As stated above, our focus is also on a Maxwellian gas for which the collision kernel does not depend on the relative velocity $|v - v_1|$, i.e., $q(v - v_1, \sigma) = q(\sigma)$, but the algorithm below can be modified to apply to a general set of collision kernels [10,40]. Under this assumption, (13) can be rewritten in the form of

$$\frac{\partial f}{\partial t} = [P(f, f) - \mu f], \tag{14}$$

where $\rho = \int_{\mathbb{R}^3} f dv$, $\mu = \rho \int_{\mathcal{S}^2} q(\sigma)d\sigma$, and

$$P(f, f) = \int_{\mathbb{R}^3}\int_{\mathcal{S}^2} q(\sigma) f' f'_1 d\sigma dv_1.$$

We remark that $f^\rho = \frac{1}{\rho} f$ is a probability density in the velocity space for any given t. It also follows the Boltzmann equation (13) with a scaled collision kernel. Without loss of generality, we regard $f = f^\rho$ as a probability density function hereafter.

In the DSMC method, we consider a set of N velocities evolving in time due to collisions whose distribution can be described by the probability distribution function f in (13). We divide time interval $[0, T]$ into M number of sub-intervals of size Δt. At the k-th time interval, the particle velocities are represented as

$$V_k = \{v_1, \ldots, v_N\}(t_k) \tag{15}$$

and we denote the i-th velocity particle in V_k as $v_i(t_k)$ or $v_{k,i}$. The forward Euler scheme applied to (14) gives

$$f_{k+1} = (1 - \mu\Delta t) f_k + \mu\Delta t \frac{P(f_k, f_k)}{\mu}, \tag{16}$$

where $f_k = f(v, k\Delta t)$. If additionally we discretize the velocity distribution at time $t_k = k\Delta t$ by the velocity particles V_k defined in (15), which is to say

$$f_k = f(v, t_k) = f(v, k\Delta t) \approx \frac{1}{N}\sum_{i=1}^{N} \delta(v - v_{k,i}), \quad k = 1, 2, \ldots, M,$$

we can interpret (16) in terms of probability, which is the core idea of the method.

At time t_k, a particle with velocity $v_{k,i}$ will not collide with probability $(1 - \mu\Delta t)$, and it will collide with another velocity particle with probability $\mu\Delta t$, according to the collision law described by $P(f_k, f_k)(v)$. Nanbu proposed an algorithm based on this probabilistic interpretation [38], and later its convergence was proved by Babovsky and Illner [6]. One can view Algorithm 1 as the realization of DSMC with the forward Euler scheme (16).

Remark 1. The decomposition (14) is convenient to illustrate the core idea of DSMC but does not apply to all collision models. The acceptance-rejection method using "virtual collisions" is required to sample from a more general cross section [15].

Algorithm 1 Nanbu–Babovsky algorithm for Maxwellian molecules.

1: Compute the initial velocity of particles based on the given initial condition, $V_0 = \{v_{0,1}, \dots, v_{0,N}\}$. Set $N_c = \lceil N\Delta t\mu \rceil$ and $M = T/\Delta t$ for final time T.
2: **for** $k = 1$ to M **do**
3: Given the velocity of particles from the previous time step, V_{k-1}.
4: Select $N_c/2$ collision pairs (i, j) uniformly among all possible pairs without replacement.
5: For those selected pairs, perform the collision between $v_{k,i}$ and $v_{k,j}$ based on (2); obtain the post-collision velocity $v'_{k,i}$ and $v'_{k,j}$.
6: Set $v_{k+1,i} = v'_{k,i}$ and $v_{k+1,j} = v'_{k,j}$.
7: Set $v_{k+1,i} = v_{k,i}$ for all particles that have not collided.
8: **end for**

3. Continuous adjoint Boltzmann equations

We consider an idealized optimization problem for the spatially homogeneous Boltzmann equation (13). The initial condition is

$$f(v,0) = f_0(v;\alpha), \tag{17}$$

in which f_0 is the prescribed initial data depending on a parameter α. We aim to find α which optimizes the objective function at time $t = T$,

$$J_1(\alpha) = \int_{\mathbb{R}^3} r(v) f(v,T) dv. \tag{18}$$

When the number of unknowns is large, which is the dimensionality of α in our case, the adjoint-state method is necessary as an efficient numerical method for computing the gradient of a function or operator in a numerical optimization problem [9,16]. It has applications in geophysics [20], seismic imaging [41] and general inverse problems [36]. It is also the theoretical foundation that gave rise to the back-propagation method in the 1990s for neural networks and machine learning [32]. One outstanding advantage of the adjoint-state method is that the number of PDE solves is independent of the dimension of the parameter for which the gradient needs to be calculated.

3.1. Derivation of the continuous adjoint equation

Following the adjoint-state method, we aim to derive the adjoint equations for our optimization problem starting with the Lagrangian

$$J = \underbrace{\int_{\mathbb{R}^3} r(v) f(v,T) dv}_{J_1} + \underbrace{\int_{\mathbb{R}^3} \kappa(v)(f(v,0) - f_0(v;\alpha)) dv}_{J_2} + \underbrace{\int_0^T \int_{\mathbb{R}^3} \gamma(v,t)(\partial_t f(v,t) - Q(f,f)) dv dt}_{J_3}$$

in which $\kappa(v)$ in J_2 is a Lagrange multiplier that enforces the initial condition for any $v \in \mathbb{R}^3$, and $\gamma(v,t)$ in J_3 is a Lagrange multiplier that enforces the Boltzmann equation for any v and t.

We remark that $f(v,t)$ is any function here, and its dependence on the Boltzmann equation and the initial condition is imposed through Lagrange multipliers.

We rewrite J_3 to calculate its Fréchet derivative with respect to f.

$$J_3 = \underbrace{\int_0^T \int_{\mathbb{R}^3} \gamma(v,t) \partial_t f(v,t) dv dt}_{J_{31}} + \underbrace{(-1) \int_0^T \int_{\mathbb{R}^3} \gamma(v,t) Q(f,f)) dv dt}_{J_{32}}.$$

After integrating by parts, we have

$$J_{31} = -\int_0^T \int_{\mathbb{R}^3} f(v,t) \partial_t \gamma(v,t) dv dt + \int_{\mathbb{R}^3} \gamma(v,T) f(v,T) dv - \int_{\mathbb{R}^3} \gamma(v,0) f(v,0) dv,$$

and for $0 < t < T$ the Fréchet derivatives are

$$\frac{\delta J_{31}}{\delta f(v,t)} = -\partial_t \gamma(v,t), \quad \frac{\delta J_{31}}{\delta f(v,T)} = \gamma(v,T), \quad \frac{\delta J_{31}}{\delta f(v,0)} = -\gamma(v,0).$$

The calculation of $\frac{\delta J_{32}}{\delta f}$ follows the derivation of collisional invariants for the Boltzmann equation [19]. It follows, using the symmetries (3), that

$$J_{32} = -\int_0^T \int_{\mathbb{R}^3} \gamma(v,t) Q(f,f) dv dt = -\int_0^T \int_{\mathbb{R}^3} \int_{\mathbb{R}^3} \int_{\mathcal{S}^2} \gamma(f_1' f' - f_1 f) q d\sigma dv_1 dv dt$$

$$= -\int_0^T \int_{\mathbb{R}^3} \int_{\mathbb{R}^3} \int_{\mathcal{S}^2} (\gamma' - \gamma) f_1 f q d\sigma dv_1 dv dt \quad \text{(by switching } v, v' \text{ in } \gamma f_1' f')$$

$$= -\int_0^T \int_{\mathbb{R}^3} \int_{\mathbb{R}^3} \int_{\mathcal{S}^2} (\gamma_1' - \gamma_1) f_1 f q d\sigma dv_1 dv dt \quad \text{(by switching } v, v_1)$$

$$= -\frac{1}{2} \int_0^T \int_{\mathbb{R}^3} \int_{\mathbb{R}^3} \int_{\mathcal{S}^2} (\gamma_1' + \gamma' - \gamma_1 - \gamma) f_1 f q d\sigma dv_1 dv dt,$$

where $\gamma, \gamma_1, \gamma'$ and γ_1' are shorthand notations for $\gamma(v,t), \gamma(v_1,t), \gamma(v',t)$ and $\gamma(v_1',t)$.

Now we perturb f by an amount δf and investigate the resulting change in J_{32}. We obtain the first-order variation:

$$\delta J_{32} = -\frac{1}{2} \int_0^T \int_{\mathbb{R}^3} \int_{\mathbb{R}^3} \int_{\mathcal{S}^2} (\gamma_1' + \gamma' - \gamma_1 - \gamma)(f_1 \delta f + f \delta f_1) q d\sigma dv_1 dv dt$$

$$= -\int_0^T \int_{\mathbb{R}^3} \int_{\mathbb{R}^3} \int_{\mathcal{S}^2} (\gamma_1' + \gamma' - \gamma_1 - \gamma) f_1 \delta f q d\sigma dv_1 dv dt.$$

Thus, the Fréchet derivative of J_{32} with respect to f is

$$\frac{\delta J_{32}}{\delta f} = -\int_{\mathbb{R}^3} \int_{\mathcal{S}^2} (\gamma_1' + \gamma' - \gamma_1 - \gamma) f_1 q d\sigma dv_1 = -L^*[f]\gamma,$$

where the second equality follows (11). Together with all the other functional derivatives, we obtain the following equations:

$$\frac{\delta J}{\delta f(v,t)} = -\partial_t \gamma - L^*[f]\gamma,$$
$$\frac{\delta J}{\delta f(v,T)} = \gamma(v,T) + r(v),$$
$$\frac{\delta J}{\delta f(v,0)} = -\gamma(v,0) + \kappa(v),$$
$$\frac{\partial J}{\partial \alpha} = -\int_{\mathbb{R}^3} \kappa(v) \partial_\alpha f_0(v;\alpha) dv.$$

Based on the first-order necessary condition for optimality, we set these variations to zero to obtain the adjoint equation for the adjoint variable γ:

$$-\partial_t \gamma = L^*[f]\gamma, \tag{19a}$$
$$\gamma(v,T) = -r(v). \tag{19b}$$

The adjoint equation evolves backward in time from T to 0, thus the final condition (19b) is given instead of the initial condition. After time evolution, we can use $\gamma(v,0)$ to eliminate $\kappa(v)$ and obtain the gradient of the objective function with respect to parameter α:

$$\partial_\alpha J = -\int_{\mathbb{R}^3} \gamma(v,0) \partial_\alpha f_0(v;\alpha) dv. \tag{20}$$

The continuous adjoint equation (19a) is an integro-differential equation which shares significant similarities with the linearized Boltzmann Equation [18]. We will further analyze and discuss the adjoint system in Section 5.

Given an initial guess α_0 for the model parameter, the first step is to solve the forward equations (13) numerically with initial condition (17) for f. A common choice is the DSMC method because of the nonlinear collision operator. The second

step is to solve the adjoint equation (19a) numerically with the final condition (19b) for the adjoint variable γ. One can solve the adjoint equation using a finite difference method in t and grid-based quadrature in v, but this is computationally difficult because v is three dimensional, and the integral at each value of v is five-dimensional. In Section 3.2, we propose Algorithm 2 as an efficient Monte Carlo type method for solving (19a). In the third step, we compute the gradient $\partial_\alpha J$ based on (20) using the numerical solutions from the first two steps. Numerical examples and comparisons are presented in Section 6. One can then update α iteratively using a gradient-based optimization algorithm to find the optimal parameter. This is the "optimize-then-discretize" (OTD) approach to solve the PDE-constrained optimization numerically.

We remark that the entire derivation above is unrelated to the dimensionality of the model parameter α. In practice, α can be, for example, a function of the velocity domain. Hence, an accurate numerical discretization of the parameter can contain thousands of variables. It is then infeasible to obtain the gradient by numerical differentiation. On the other hand, using the adjoint-state method, the cost of evaluating the gradient once is equivalent to only one forward solve of the Boltzmann equation (13) and one (backward) solve of the adjoint equation (19a), independent of the number of variables in α. The adjoint-state method is an extremely efficient tool for large-scale optimization problems.

3.2. A particle method for the continuous adjoint Boltzmann equation

The adjoint DSMC method presented in Section 4 is the main result of this work and the most efficient and accurate that we have found for solving the adjoint problem. Nevertheless, the particle method described in this section may be of independent interest. It is based on a formal derivation with several steps that are not fully justified, but are validated by the numerical results in Section 6.

The continuous adjoint equation (19a) is based on the solution to the forward equation $f(v,t), t \in [0,T]$. Therefore, we assume that the state (forward) equation (13) has been solved numerically by the DSMC method, where the choice of collision times, collision partners, and collision angles during the simulation are all stored in memory. Based on the solution from the forward DSMC simulation, we will (approximately) solve the equation (19a) for $\gamma(v,t)$ backward in time t with the "final data" (19b).

According to (13) and (19a), we have

$$\partial_t(\gamma f) = \gamma\,\partial_t f + f\,\partial_t\gamma = \iint (f'f_1' - ff_1)\gamma q d\sigma dv_1 - \iint (\gamma_1' + \gamma' - \gamma_1 - \gamma) f_1 f q d\sigma dv_1.$$

We then multiply both sides by a time-independent test function $\psi(v)$ and integrate over the velocity domain. The left-hand side becomes

$$\int_{\mathbb{R}^3} \psi\,\partial_t(\gamma f) dv = \partial_t \left(\int_{\mathbb{R}^3} \psi \gamma f dv \right). \tag{21}$$

Using the symmetries (3), the right-hand side becomes

$$\begin{aligned}
&\iiint \psi \left\{ (f'f_1' - ff_1)\gamma - (\gamma_1' + \gamma' - \gamma_1 - \gamma) f_1 f \right\} q d\sigma dv_1 dv \\
&= \frac{1}{2} \iiint \left(\psi_1'\gamma_1' + \psi'\gamma' - \psi_1\gamma_1 - \psi\gamma \right) f_1 f q d\sigma dv_1 dv - \frac{1}{2} \iiint (\psi + \psi_1)(\gamma_1' + \gamma' - \gamma_1 - \gamma) f_1 f q d\sigma dv_1 dv \\
&= \frac{1}{2} \iiint \left\{ \psi_1'\gamma_1' + \psi'\gamma' - \psi_1(\gamma_1' + \gamma' - \gamma) - \psi(\gamma_1' + \gamma' - \gamma_1) \right\} f_1 f q d\sigma dv_1 dv.
\end{aligned} \tag{22}$$

Combining both (21) and (22), we discretize the time domain of the equation using the Euler scheme in t. Consider a time interval $[t_k, t_{k+1}]$ during which there is a collision of v and v_1 at $t = t_k$ to produce v' and v_1' at $t = t_{k+1}$.

As in the (forward) DSMC Algorithm 1, $f = f(v,t)$ and $f_1 = f(v_1,t)$ in (22) should be evaluated at t_k since v and v_1 are independent before the collision so that the product ff_1 is the joint density function for v and v_1.

On the other hand, the choice of time t at which to evaluate $\gamma(v,t)$ is a new issue and not so clearcut. We choose to evaluate $\gamma(v,t)$ at t_{k+1} based on the following considerations: First, the data for the adjoint variable comes at the final time (19b), so that (19a) should be solved backward in time, and it is most natural for the right-hand side (22) to be evaluated at the final time t_{k+1} of the time interval. Second, since γ is solved backward in time, we expect that $\gamma(v,t_{k+1})$ is independent of $f(v,t_k)$, so that it is reasonable to sample using these values. Third, our numerical computations in Section 6 verify that this choice leads to a correct result.

Based on these choices, we take $\gamma(v,t) = \gamma_{k+1}(v)$ at $t = t_{k+1}$ and $f(v,t) = f_k(v)$ at $t = t_k$ in (22) to obtain (with ψ independent of t)

$$\int_{\mathbb{R}^3} \psi\gamma_{k+1} f_{k+1} dv - \int_{\mathbb{R}^3} \psi\gamma_k f_k dv \approx \frac{\Delta t}{2} \iiint (\psi_1'\gamma_{k+1,1}' + \psi'\gamma_{k+1}') f_k(v_1) f_k(v) q(\sigma) d\sigma dv_1 dv$$

$$-\frac{\Delta t}{2}\iiint \psi_1(\gamma'_{k+1,1}+\gamma'_{k+1}-\gamma_{k+1})f_k(v_1)f_k(v)q(\sigma)d\sigma dv_1 dv$$
$$-\frac{\Delta t}{2}\iiint \psi(\gamma'_{k+1,1}+\gamma'_{k+1}-\gamma_{k+1,1})f_k(v_1)f_k(v)q(\sigma)d\sigma dv_1 dv. \tag{23}$$

Without loss of generality, we assume $\rho=\int f dv=1$ and $\mu=\rho\int q(\sigma)d\sigma=1$. Then $f(v)$ is a probability density in $\mathbb{R}^3$, and $F(\sigma,v,v_1)=f_k(v)f_k(v_1)q(\sigma)$ is a probability density function in the product space $\mathcal{S}^2\times\mathbb{R}^3\times\mathbb{R}^3$. We apply Monte Carlo quadrature to approximate the integrals on both sides of (23), using the velocities from V_k and V_{k+1} (defined as in (15)) and values of σ that were selected in the forward DSMC calculation. Similar to (15), we represent the adjoint variables as

$$\Gamma_{k+1}=\{\hat\gamma_1,\dots,\hat\gamma_i,\dots\hat\gamma_N\}(t_{k+1}).$$

Both sides of the resulting Monte Carlo sums for (23) are nonzero only for velocities that undergo collisions. For the Monte Carlo quadrature we have $N_c/2=\lceil\mu\Delta t N\rceil/2$ collision pairs.

The j-th collision between velocities $v_j, v_{j_1}\in V_k$, with collision parameters σ_j, results in velocities $v'_j, v'_{j_1}\in V_{k+1}$. The corresponding values of γ are denoted as $\hat\gamma,\hat\gamma_1\in\Gamma_k$ and $\hat\gamma',\hat\gamma'_1\in\Gamma_{k+1}$.

On the left-hand side of (23), the term γ_{k+1} is represented by $\hat\gamma',\hat\gamma'_1$ in Γ_{k+1} and the term γ_k is represented by $\hat\gamma,\hat\gamma_1\in\Gamma_k$.

On the right-hand side of (23), the terms $\gamma'_{k+1},\gamma'_{k+1,1}$ are represented by $\hat\gamma',\hat\gamma'_1\in\Gamma_{k+1}$, but the terms $\gamma_{k+1},\gamma_{k+1,1}$ are at $t=t_{k+1}$ and correspond to velocities $v,v_1\notin V_{k+1}$, so that the corresponding $\hat\gamma$ values are not in Γ_{k+1}. We will instead use the values of the continuous function $\gamma(v,t_{k+1})$ and $\gamma(v_1,t_{k+1})$.

The resulting quadrature approximation for (23) by Monte Carlo sampling is

$$\begin{aligned}&\sum_{j=1}^{N_c/2}\psi'_{j_1}\hat\gamma_{j_1}(t_{k+1})+\psi'_j\hat\gamma_j(t_{k+1})-\psi_{j_1}\hat\gamma_{j_1}(t_k)-\psi_j\hat\gamma_j(t_k) \\ \approx&\sum_{j=1}^{N_c/2}\psi'_{j_1}\hat\gamma_{j_1}(t_{k+1})+\psi'_j\hat\gamma_j(t_{k+1})-\psi_{j_1}\left(\hat\gamma_{j_1}(t_{k+1})+\hat\gamma_j(t_{k+1})-\gamma(v_j,t_{k+1})\right) \\ -&\ \psi_j\left(\hat\gamma_{j_1}(t_{k+1})+\hat\gamma_j(t_{k+1})-\gamma(v_{j_1},t_{k+1})\right),\end{aligned} \tag{24}$$

where $\psi_j=\psi(v_j)$, $\psi_{j_1}=\psi(v_{j_1})$, $\psi'_j=\psi(v'_j)$ and $\psi'_{j_1}=\psi(v'_{j_1})$.

The values of the continuous adjoint function in (24) can be expressed in terms of a conditional expectation:

$$\gamma(v,t)=\mathbb{E}[\hat\gamma_i(t)|v_i=v].$$

Since ψ is an arbitrary test function, and (24) holds for any ψ, we can match the coefficients and obtain a Monte Carlo type numerical scheme for solving $\gamma(v,t)$. If velocity particle v_j did not collide at time t_k,

$$\hat\gamma_j(t_k)=\hat\gamma_j(t_{k+1}). \tag{25}$$

If v_j and v_{j_1} are a collision pair at time t_k,

$$\hat\gamma_j(t_k)=\hat\gamma_j(t_{k+1})+\hat\gamma_{j_1}(t_{k+1})-\mathbb{E}[\hat\gamma_i(t_{k+1})|v_i=v_{j_1}], \tag{26a}$$
$$\hat\gamma_{j_1}(t_k)=\hat\gamma_j(t_{k+1})+\hat\gamma_{j_1}(t_{k+1})-\mathbb{E}[\hat\gamma_i(t_{k+1})|v_i=v_j]. \tag{26b}$$

Following [34,45], one may approximate the expectation $\mathbb{E}[\hat\gamma_i(t_{k+1})|v_i=v_j]$ by numerical interpolation as the following:

$$\hat\gamma(v_j,t_{k+1})=\mathbb{E}[\hat\gamma_i(t_{k+1})|v_i=v_j]\approx\frac{1}{L}\sum_{l=1}^{L}\omega(v_j-v_l)\hat\gamma_l(t_{k+1}) \tag{27}$$

for appropriately chosen interpolation coefficients $\omega(v)$.

Given the final condition (19b), the continuous adjoint equation (19a) can be solved numerically following Algorithm 2.

Remark 2. We regard the numerical method in Algorithm 2 as the DSMC-type scheme for the continuous adjoint equation, depending on a solution of (13) by the forward DSMC method following Algorithm 1. The gradient of the objective function (20) can be computed using the values of $\gamma(v,t)$ at $t=0$, i.e., Γ_0. Numerical examples are shown in Section 6.3.

Algorithm 2 Algorithm for solving the continuous adjoint equation (19).

Given the final-time velocity particles V_M in the forward DSMC and the final condition (19b), set $\hat{\gamma}_i(T) = -r(v_{M,i}),\ i = 1, \dots, N$. Obtain Γ_M.
for $k = M-1$ to 0 **do**
  Given Γ_{k+1} from the previous iteration and V_k from the forward DSMC.
  if $v_j \in V_k$ did not collide at t_k **then**
    Set $\hat{\gamma}_j(t_k) = \hat{\gamma}_j(t_{k+1})$.
  else if $v_j, v_{j_1} \in V_k$ collided at t_k **then**
    Approximate $\mathbb{E}[\hat{\gamma}_i(t_{k+1})|v_i = v_j]$ and $\mathbb{E}[\hat{\gamma}_i(t_{k+1})|v_i = v_{j_1}]$ following (27).
    Set $\hat{\gamma}_j(t_k)$ and $\hat{\gamma}_{j_1}(t_k)$ following (26).
  end if
  Obtain Γ_k.
end for

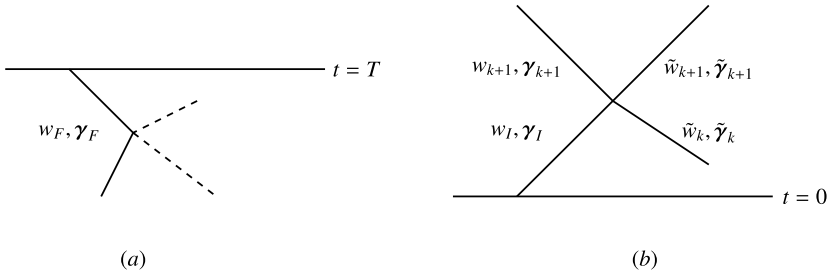

Fig. 2. (*a*) Collision for velocity particles at the final time $t = T$; (*b*) Collision for velocity particles at the initial time $t = 0$.

4. Adjoint DSMC for Boltzmann equation

In this section, we derive the adjoint system based on the DTO approach. We first rewrite the objective functions and the constraints based on the DSMC method, to which the Lagrange multiplier method will be applied.

Following Algorithm 1, we discretize the time $[0, T]$ into M equal intervals $[t_k, t_{k+1}]$ and denote $v_{k,i}$ to be the velocity of the i-th particle at time t_k, for $1 \le i \le N$ and $0 \le k \le M$. We also denote $v_{I,i}$ to be the velocity of the i-th particle at the initial time $t = 0$, and $v_{F,i}$ to be the velocity of the i-th particle at the final time $t = T$. The DSMC method applied to the spatially homogeneous Boltzmann equation consists of a sequence of particle collisions in sequential order, and the time step Δt plays no role. So we will choose Δt in whatever way is most convenient for the exposition. In this section, Δt will be chosen small enough that there is at most a single collision in each time step.

Consider the new objective function $\mathcal{J}$ defined as

$$\mathcal{J} = \underbrace{\frac{1}{N}\sum_{i=1}^{N} r(v_{F,i})}_{\mathcal{J}_1} + \underbrace{\frac{1}{N}\sum_{i=1}^{N} \boldsymbol{\gamma}_{I,i} \cdot (v_{I,i} - v_{0,i}(\alpha))}_{\mathcal{J}_2} + \underbrace{\frac{1}{N}\sum_{k=1}^{M}\sum_{i=1}^{N} \boldsymbol{\gamma}_{k,i} \cdot (v_{k+1,i} - v'_{k,i})}_{\mathcal{J}_3}. \tag{28}$$

Here, $\mathcal{J}_1$ is the Monte Carlo quadrature of the objective function (18) by particle velocities $\{v_{F,i}\}_{i=1}^N$, $\mathcal{J}_2$ is the constraint on the DSMC initial condition $v = v_0(\alpha)$ using the Lagrange multiplier $\{\boldsymbol{\gamma}_{I,i}\}_{i=1}^N$, and $\mathcal{J}_3$ is the constraint that enforces the binary collision law (2) using the Lagrange multiplier $\boldsymbol{\gamma}_{k,i}$ for each particle i at the k-th time interval. In particular, $v'_{k,i}$ represents the post-collision velocity of particle i if it participates in the collision at the k-th time interval. Otherwise, $v'_{k,i} = v_{k,i}$, which means the particle velocity remains the same at the $(k+1)$-th time interval.

Again, we regard that $\{v_{k,i}\}$ is a general set of velocities, and its dependence on the collision rules (2) is imposed through the Lagrange multipliers. The following variation is easily calculated:

$$\partial_\alpha \mathcal{J} = -\frac{1}{N}\sum_{i=1}^{N} \boldsymbol{\gamma}_{I,i} \cdot \partial_\alpha v_{0,i}(\alpha) \tag{29}$$

We proceed to derive the equations in the backward time direction from the final time $t = T$ to the initial time $t = 0$. First, we consider the equations that come from the derivative of $\mathcal{J}$ with respect to $v_{F,i}$, which are derived in Section 4.1. Second, we consider collisions that occur at a time t where $0 < t < T$. The derivatives of $\mathcal{J}$ with respect to the velocities in these collisions are derived in Section 4.2.

4.1. Velocities at the final time for DSMC

We denote a particular particle velocity at the final time as $w_F = v_{F,i}$ for some i, and the corresponding value of its dual variable is $\boldsymbol{\gamma}_F = \boldsymbol{\gamma}_{F,i}$, as shown in Fig. 2(a). The velocity w_F occurs in the objective function $\mathcal{J}$ at two places. First,

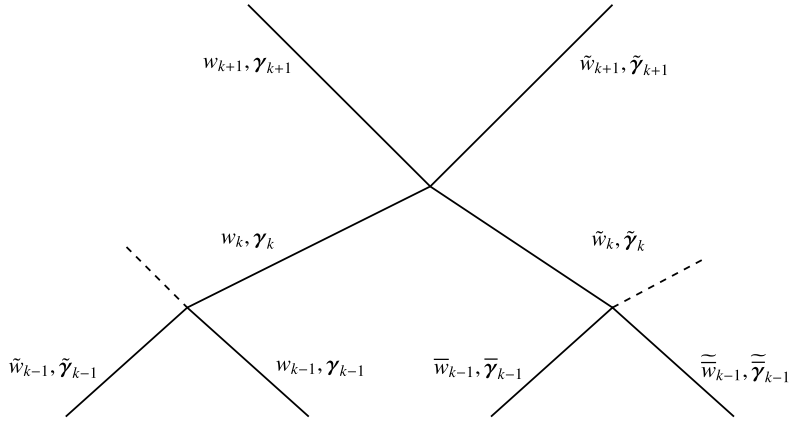

Fig. 3. Collision for velocity particles at intermediate time.

it directly appears in the term $r(v_{F,i})$ in $\mathcal{J}_1$. Second, it occurs in the last collision that involves $v_{F,i}$, through the term $\boldsymbol{\gamma}_{F,i} \cdot (v_{F,i} - v'_{k,i})$ in $\mathcal{J}_3$. In this term, $v'_{k,i}$ is a velocity produced by the collision, and is a function of the two velocities before the collision, but $v'_{k,i}$ does not depend on $v_{F,i}$ here. Therefore,

$$\partial_{w_F} \mathcal{J} = \frac{1}{N} \partial_{w_F} r(w_F) + \frac{1}{N} \partial_{w_F} \left(\boldsymbol{\gamma}_F \cdot (w_F - v'_{k,i}) \right) = \frac{1}{N} \partial_v r(w_F) + \frac{1}{N} \boldsymbol{\gamma}_F.$$

The resulting equation comes from the first-order optimality $\partial_{w_F} \mathcal{J} = 0$:

$$\boldsymbol{\gamma}_F = -\partial_v r(w_F). \tag{30}$$

Equation (30) provides the starting values for solving the adjoint variable $\boldsymbol{\gamma}$ backward in time.

4.2. Collisions away from the final time for DSMC

According to the DSMC method, with V_k denoting the set of velocities in the k-th time interval as in (15), the collision operator C takes V_k to $V_{k+1} = C[V_k]$ as follows: We first choose indices i and j, and denote the corresponding velocities as $(w_k, \tilde{w}_k) = (v_{k,i}, v_{k,j})$ from V_k and also choose a unit vector σ_k. Here, we assume the i-th and the j-th particles collide at the k-th time interval. Then C takes $(w_k, \tilde{w}_k)$ to

$$(w_{k+1}, \tilde{w}_{k+1}) = (w'_k, \tilde{w}'_k)$$

where

$$w'_k = \frac{1}{2}((w_k + \tilde{w}_k) + |w_k - \tilde{w}_k|\sigma_k), \tag{31a}$$

$$\tilde{w}'_k = \frac{1}{2}((w_k + \tilde{w}_k) - |w_k - \tilde{w}_k|\sigma_k). \tag{31b}$$

Note that the resulting velocities $(w_{k+1}, \tilde{w}_{k+1})$ correspond to $(v'_{k,i}, v'_{k,j})$.

On the other hand, we denote w_{k-1} and $\tilde{w}_{k-1}$ to be the particle velocities whose collision produces w_k. Similarly, $\tilde{w}_k$ is produced by particle velocities $\overline{w}_{k-1}$ and $\widetilde{\overline{w}}_{k-1}$. Particle velocities

$$w_{k-1},\ \tilde{w}_{k-1},\ \overline{w}_{k-1},\ \widetilde{\overline{w}}_{k-1},\ w_k,\ \tilde{w}_k, w_{k+1},\ \tilde{w}_{k+1}$$

correspond to the dual (adjoint) variables

$$\boldsymbol{\gamma}_{k-1},\ \tilde{\boldsymbol{\gamma}}_{k-1},\ \overline{\boldsymbol{\gamma}}_{k-1},\ \widetilde{\overline{\boldsymbol{\gamma}}}_{k-1},\ \boldsymbol{\gamma}_k,\ \tilde{\boldsymbol{\gamma}}_k,\ \boldsymbol{\gamma}_{k+1},\ \tilde{\boldsymbol{\gamma}}_{k+1},$$

respectively, as shown in Fig. 3, which schematically depicts the three collisions that involve particles w_k and $\tilde{w}_k$.

Next, we recall the variational quantity $\mathcal{J}_3$ for collisions, with dual quantities $\boldsymbol{\gamma}$, as

$$\mathcal{J}_3 = \frac{1}{N} \sum_k \sum_i \boldsymbol{\gamma}_{k,i} \cdot (v_{k+1,i} - v'_{k,i}).$$

We use $\mathcal{J}_{3k}$ to denote the terms in $\mathcal{J}_3$ that involve w_k or $\tilde{w}_k$:

$$\mathcal{J}_{3k} = \frac{1}{N}\left\{\boldsymbol{\gamma}_k \cdot (w_k - w'_{k-1}) + \tilde{\boldsymbol{\gamma}}_k \cdot (\tilde{w}_k - \overline{w}'_{k-1}) + \boldsymbol{\gamma}_{k+1} \cdot (w_{k+1} - w'_k) + \tilde{\boldsymbol{\gamma}}_{k+1} \cdot (\tilde{w}_{k+1} - \tilde{w}'_k)\right\}.$$

Note that w'_{k-1} is a function only of w_{k-1} and $\tilde{w}_{k-1}$, and that $\overline{w}'_{k-1}$ is a function only of $\overline{w}_{k-1}$ and $\widetilde{\overline{w}}_{k-1}$. Consequently, w'_{k-1} and $\overline{w}'_{k-1}$, as well as w_{k+1} and $\tilde{w}_{k+1}$, do not depend on w_k and $\tilde{w}_k$. It follows that

$$\partial_{w_k}\mathcal{J}_3 = \partial_{w_k}\mathcal{J}_{3k} = \frac{1}{N}\left(\boldsymbol{\gamma}_k - \boldsymbol{\gamma}_{k+1} \cdot \partial_{w_k} w'_k - \tilde{\boldsymbol{\gamma}}_{k+1} \cdot \partial_{w_k} \tilde{w}'_k\right), \tag{32a}$$

$$\partial_{\tilde{w}_k}\mathcal{J}_3 = \partial_{\tilde{w}_k}\mathcal{J}_{3k} = \frac{1}{N}\left(\tilde{\boldsymbol{\gamma}}_k - \boldsymbol{\gamma}_{k+1} \cdot \partial_{\tilde{w}_k} w'_k - \tilde{\boldsymbol{\gamma}}_{k+1} \cdot \partial_{\tilde{w}_k} \tilde{w}'_k\right). \tag{32b}$$

These derivatives can be calculated from (31) to get

$$\begin{aligned}
\boldsymbol{\gamma} \cdot \partial_{w_k} w'_k &= \frac{1}{2}\left(\boldsymbol{\gamma} + \boldsymbol{\gamma} \cdot \sigma_k (w_k - \tilde{w}_k)\hat{}\right),\\
\boldsymbol{\gamma} \cdot \partial_{\tilde{w}_k} w'_k &= \frac{1}{2}\left(\boldsymbol{\gamma} - \boldsymbol{\gamma} \cdot \sigma_k (w_k - \tilde{w}_k)\hat{}\right),\\
\boldsymbol{\gamma} \cdot \partial_{w_k} \tilde{w}'_k &= \frac{1}{2}\left(\boldsymbol{\gamma} - \boldsymbol{\gamma} \cdot \sigma_k (w_k - \tilde{w}_k)\hat{}\right),\\
\boldsymbol{\gamma} \cdot \partial_{\tilde{w}_k} \tilde{w}'_k &= \frac{1}{2}\left(\boldsymbol{\gamma} + \boldsymbol{\gamma} \cdot \sigma_k (w_k - \tilde{w}_k)\hat{}\right),
\end{aligned}$$

where $x\hat{} = x/|x|$ denotes the unit vector along the direction of x. Therefore,

$$\begin{aligned}
\partial_{w_k}\mathcal{J}_3 &= \frac{1}{N}\left\{\boldsymbol{\gamma}_k - \frac{1}{2}(\boldsymbol{\gamma}_{k+1} + \tilde{\boldsymbol{\gamma}}_{k+1}) - \frac{1}{2}(\boldsymbol{\gamma}_{k+1} - \tilde{\boldsymbol{\gamma}}_{k+1}) \cdot \sigma_k (w_k - \tilde{w}_k)\hat{}\right\},\\
\partial_{\tilde{w}_k}\mathcal{J}_3 &= \frac{1}{N}\left\{\tilde{\boldsymbol{\gamma}}_k - \frac{1}{2}(\boldsymbol{\gamma}_{k+1} + \tilde{\boldsymbol{\gamma}}_{k+1}) + \frac{1}{2}(\boldsymbol{\gamma}_{k+1} - \tilde{\boldsymbol{\gamma}}_{k+1}) \cdot \sigma_k (w_k - \tilde{w}_k)\hat{}\right\}.
\end{aligned}$$

Setting these two partial derivatives to 0, gives the equations

$$\boldsymbol{\gamma}_k = \frac{1}{2}(\boldsymbol{\gamma}_{k+1} + \tilde{\boldsymbol{\gamma}}_{k+1}) + \frac{1}{2}(\boldsymbol{\gamma}_{k+1} - \tilde{\boldsymbol{\gamma}}_{k+1}) \cdot \sigma_k (w_k - \tilde{w}_k)\hat{}, \tag{33a}$$

$$\tilde{\boldsymbol{\gamma}}_k = \frac{1}{2}(\boldsymbol{\gamma}_{k+1} + \tilde{\boldsymbol{\gamma}}_{k+1}) - \frac{1}{2}(\boldsymbol{\gamma}_{k+1} - \tilde{\boldsymbol{\gamma}}_{k+1}) \cdot \sigma_k (w_k - \tilde{w}_k)\hat{}. \tag{33b}$$

Moreover, (33) can be rewritten using the operator notation (7) from Section 2.2 as

$$\begin{pmatrix}\boldsymbol{\gamma}_{k+1}\\ \tilde{\boldsymbol{\gamma}}_{k+1}\end{pmatrix} = A(\sigma_k, \alpha_k)\begin{pmatrix}\boldsymbol{\gamma}_k\\ \tilde{\boldsymbol{\gamma}}_k\end{pmatrix}, \quad \begin{pmatrix}\boldsymbol{\gamma}_k\\ \tilde{\boldsymbol{\gamma}}_k\end{pmatrix} = B(\sigma_k, \alpha_k)\begin{pmatrix}\boldsymbol{\gamma}_{k+1}\\ \tilde{\boldsymbol{\gamma}}_{k+1}\end{pmatrix}, \tag{34}$$

in which $\alpha_k = (w_k - \tilde{w}_k)\hat{}$.

Note that (33) for $\boldsymbol{\gamma}_k$ and $\tilde{\boldsymbol{\gamma}}_k$ does not involve any variables at time interval $k-1$. It follows that (33) is valid for values of $\boldsymbol{\gamma}_I = \boldsymbol{\gamma}_k$ and $\tilde{\boldsymbol{\gamma}}_k$ corresponding to a collision in which one (or both) of the velocities $w_I = w_k$ started at $t = 0$, as depicted in Fig. 2(b).

The adjoint DSMC equations (33a)-(33b), which we derive following the DTO approach, are the DSMC analog of the dual equation (19a), and they are one of the main results of this paper. The evolution equations for the dual variable $\boldsymbol{\gamma}$ are solved backward in time, from $(k+1)$-th time interval to the k-th time interval, for each k. For each collision, these equations are solved for the dual variables corresponding to the particle velocity variables involved in that collision.

4.3. Summary for discrete adjoint system

In summary, we solve the direct equations in DSMC forward in time for the velocity particles $V = \{v_1, \ldots, v_N\}$ with the initial particle velocity sampled from (17) given the current value of α. Collisions are performed according to (2). Then the adjoint equations (33a)-(33b) for the adjoint particle $\boldsymbol{\gamma}$ are solved backward in time following Algorithm 3. At the initial time $t = 0$, we use (29) to calculate the gradient of the objective function with respect to the model parameter α, which can be used by optimization algorithms to update the α iteratively.

Like the continuous adjoint system (Section 3), by solving the direct equations and the adjoint equations only once, we obtain all components of the gradient $\partial_\alpha J$, for any value of the dimensionality of α. We remark that the choice of collision times, collision partners, and collision angles is not included in the variational principle. These parts of the DSMC method are determined externally.

The restriction to Maxwell molecules significantly simplifies the adjoint DSMC method presented in this section. Since the collision kernel q is a constant, the collision rate for a pair of velocities v and v_1 and the choice of the collision

Algorithm 3 Algorithm for solving the discrete adjoint DSMC system (33).

Given the final-time velocity particles $V_M = \{v_{F,1}, \dots, v_{F,N}\}$ from the forward DSMC, set $\boldsymbol{\gamma}_{F,j} = -\partial_v r(v_{F,j})$ for $j = 1, \dots, N$ following (30).
for $k = M-1$ to 0 **do**
  Given $\{\boldsymbol{\gamma}_{k+1,1}, \dots, \boldsymbol{\gamma}_{k+1,N}\}$ from the previous iteration and collision parameters in the forward DSMC.
  if $v_j \in V_k$ did not collide at t_k **then**
    Set $\boldsymbol{\gamma}_{k,j} = \boldsymbol{\gamma}_{k+1,j}$.
  else if $v_j, v_{j_1} \in V_k$ collided at t_k **then**
    Perform backward collision between $\boldsymbol{\gamma}_{k+1,j}$ and $\boldsymbol{\gamma}_{k+1,j_1}$ and obtain $\boldsymbol{\gamma}_{k,j}$ and $\boldsymbol{\gamma}_{k,j_1}$ following (33a) and (33b).
  end if
  Obtain $\{\boldsymbol{\gamma}_{k,1}, \boldsymbol{\gamma}_{k,2}, \dots, \boldsymbol{\gamma}_{k,N}\}$.
end for

parameter σ do not depend on the values of the velocities. This implies that variations δv and δv_1 in the velocities do not cause variations in the choice of collision pairs nor in the collision parameter, which simplifies the adjoint DSMC equation (33). This is the main reason that we restricted our attention to Maxwell molecules in this work. We believe it is possible to extend the adjoint DSMC method to non-Maxwell interactions. Extension to collision kernels $q = q(\theta)$ with angular dependence should be straightforward since, in that case, the only thing that changes is the sampling of the collision parameter σ, while it is still independent of pre-collision velocities v and v_1. Extensions to the VHS model with $q = q(|v - v_1|)$ collision kernel and other non-Maxwell interactions are more intricate since either the collision rates or the collision parameter σ will depend on pre-collision velocities. The former could be mitigated by using a different DSMC method with a constant collision rate such as [12], while the latter would produce additional terms in $\partial_{w_k} w_k'$, $\partial_{\tilde{w}_k} w_k'$, $\partial_{w_k} \tilde{w}_k'$, $\partial_{\tilde{w}_k} \tilde{w}_k'$ in (32). We leave such generalizations for future work and they are beyond the scope of this paper.

5. Relationship between the continuous adjoint and the adjoint DSMC formulation

In this section, we discuss the similarities and differences between the continuous adjoint formulation (Section 3) and the adjoint DSMC formulation (Section 4). The former is derived under the "optimize-then-discretize" (OTD) approach, while the latter is based on the "discretize-then-optimize" (DTO) approach. For a given discretization scheme, OTD and DTO may not be equivalent, as demonstrated by many examples in the literature [28,13,1,29,33]. The DTO approach is particularly preferred for problems that are otherwise unsolvable under the OTD approach, such as the heat-transfer optimization problem [7,26].

Since DTO and OTD are generally not the same, we investigate the analytical properties of the two adjoint systems. The analysis in Section 5.1 illustrates the essential role of the continuous adjoint variable $\gamma(v,t)$, as a Fréchet derivative of the objective function. We find a similar result for the discrete adjoint variable $\boldsymbol{\gamma}_{k,i}$ for DSMC in Section 5.2. Based on these results, we find a direct relationship between $\gamma(v,t)$ and $\boldsymbol{\gamma}_{k,i}$ in Section 5.3 that connects the two adjoint systems.

We remark that the derivatives in this section have slightly different meanings from the ones used before. In Section 3 and Section 4, the constraints for the state variables are not directly applied, but instead are imposed by the adjoint variables in the Lagrangian formulation. Here in Section 5, the derivatives with respect to $f(v)$ or v_k are computed with the assumption that they are directly constrained to satisfy the Boltzmann equation (13) or the DSMC equations, respectively.

5.1. Continuous adjoint variable as a Fréchet derivative

If $f(v,t)$ is a solution of the Boltzmann equation (13), then the linearized equation for a perturbed distribution $f(v,t) + \delta f(v,t)$ is

$$\partial_t \delta f(v,t) = f(v,t)\, L[f(v,t)] \left(\frac{\delta f(v,t)}{f(v,t)} \right) \tag{35}$$

using the linearized collision operator (9).

For objective function $J_1(\alpha) = \int_v r(v) f(v,T) dv$, the perturbation δf causes a perturbation δJ_1, which satisfies

$$\delta J_1 = \int_{\mathbb{R}^3} \delta f(v,t) \frac{\delta J_1}{\delta f(v,t)} dv$$

for any t because the perturbation $\delta f(t,\cdot)$ determines $\delta f(t',\cdot)$ for any $t' > t$. Since J_1 is time-independent, $\forall t \in [0,T]$, we have

$$\begin{aligned} 0 = \partial_t \delta J_1 &= \partial_t \left(\int_{\mathbb{R}^3} \delta f(v,t) \frac{\delta J_1}{\delta f(v,t)} dv \right) \\ &= \int_{\mathbb{R}^3} (\partial_t \delta f) \left(\frac{\delta J_1}{\delta f} \right) dv + \int_{\mathbb{R}^3} \delta f\, \partial_t \left(\frac{\delta J_1}{\delta f} \right) dv \end{aligned}$$

$$= \int_{\mathbb{R}^3} f\, L[f]\left(\frac{\delta f}{f}\right)\frac{\delta J_1}{\delta f}dv + \int_{\mathbb{R}^3} \delta f\, \partial_t\left(\frac{\delta J_1}{\delta f}\right)dv$$
$$= \int_{\mathbb{R}^3} f\, L^*[f]\left(\frac{\delta J_1}{\delta f}\right)\frac{\delta f}{f}dv + \int_{\mathbb{R}^3} \delta f\, \partial_t\left(\frac{\delta J_1}{\delta f}\right)dv$$
$$= \int_{\mathbb{R}^3} \delta f\left(L^*[f]\left(\frac{\delta J_1}{\delta f}\right) + \partial_t\left(\frac{\delta J_1}{\delta f}\right)\right)dv.$$

Here, we have used the chain rule, the duality between $L[f]$ and $L^*[f]$, and (35).

Since the perturbation δf is arbitrary, the following holds for any t

$$-\partial_t\left(-\frac{\delta J_1}{\delta f}\right) = L^*[f]\left(-\frac{\delta J_1}{\delta f}\right). \tag{36}$$

Note that equation (36) for $-\frac{\delta J_1}{\delta f}$ is the same as equation (19a) for the continuous adjoint function $\gamma(v,t)$. Additionally, as seen directly from the objective function, the "final" condition for (36) is $-\frac{\delta J_1}{\delta f(v,T)} = -r(v)$, the same as the "final" condition (19b) for $\gamma(v,T)$. Based on these two facts, we obtain the main result of this subsection:

$$\gamma(v,t) = -\frac{\delta J_1}{\delta f(v,t)}, \quad t \in [0,T]. \tag{37}$$

Therefore, the continuous adjoint variable $\gamma(v,t)$ is the negative Fréchet derivative of the objective function with respect to the state variable $f(v,t)$.

5.2. DSMC adjoint variable as a derivative

The analysis of the DSMC adjoint variable $\boldsymbol{\gamma}$ is a discrete version of the analysis in the previous section for the continuous adjoint variable γ. Consider small perturbations δw_k and $\delta \tilde{w}_k$ in the pre-collision velocities w_k and $\tilde{w}_k$. The resulting first-order variations in the post-collision velocities are δw_{k+1} and $\delta \tilde{w}_{k+1}$. Since no other velocities are changed at times t_k and t_{k+1} and because $\mathcal{J}_1$ is time independent, then

$$\delta\mathcal{J}_1 = \partial_{w_k}\mathcal{J}_1\cdot\delta w_k + \partial_{\tilde{w}_k}\mathcal{J}_1\cdot\delta\tilde{w}_k = \partial_{w_{k+1}}\mathcal{J}_1\cdot\delta w_{k+1} + \partial_{\tilde{w}_{k+1}}\mathcal{J}_1\cdot\delta\tilde{w}_{k+1},$$

which can be rewritten in operator form, and then by using (8), as

$$\begin{pmatrix}\partial_{w_k}\mathcal{J}_1^T & \partial_{\tilde{w}_k}\mathcal{J}_1^T\end{pmatrix}\begin{pmatrix}\delta w_k\\ \delta\tilde{w}_k\end{pmatrix} = \begin{pmatrix}\partial_{w_{k+1}}\mathcal{J}_1^T & \partial_{\tilde{w}_{k+1}}\mathcal{J}_1^T\end{pmatrix}\begin{pmatrix}\delta w_{k+1}\\ \delta\tilde{w}_{k+1}\end{pmatrix} = \begin{pmatrix}\partial_{w_{k+1}}\mathcal{J}_1^T & \partial_{\tilde{w}_{k+1}}\mathcal{J}_1^T\end{pmatrix}A(\sigma_k,\alpha_k)\begin{pmatrix}\delta w_k\\ \delta\tilde{w}_k\end{pmatrix}.$$

Since this is true for any values of δw_k and $\delta\tilde{w}_k$, it follows that

$$\begin{pmatrix}\partial_{w_k}\mathcal{J}_1^T & \partial_{\tilde{w}_k}\mathcal{J}_1^T\end{pmatrix} = \begin{pmatrix}\partial_{w_{k+1}}\mathcal{J}_1^T & \partial_{\tilde{w}_{k+1}}\mathcal{J}_1^T\end{pmatrix}A(\sigma_k,\alpha_k).$$

By taking the transpose and using $A^T = B$, then

$$\begin{pmatrix}\partial_{w_k}\mathcal{J}_1\\ \partial_{\tilde{w}_k}\mathcal{J}_1\end{pmatrix} = B(\sigma_k,\alpha_k)\begin{pmatrix}\partial_{w_{k+1}}\mathcal{J}_1\\ \partial_{\tilde{w}_{k+1}}\mathcal{J}_1\end{pmatrix}. \tag{38}$$

Equation (38) shows that $(\partial_{w_k}\mathcal{J}_1, \partial_{\tilde{w}_k}\mathcal{J}_1)$ satisfies the same back-propagation rule as (34) for $(\boldsymbol{\gamma}_k, \tilde{\boldsymbol{\gamma}}_k)$.

Recall (30) for the final condition for the adjoint DSMC particles,

$$\boldsymbol{\gamma}_F = -\partial_v r(w_F) = -N\partial_{w_F}\mathcal{J}_1, \tag{39}$$

in which the second equation comes from the definition of $\mathcal{J}_1$ in (28). With the same final condition (39) (up to a constant scaling $-N$), and the same back-propagating rule (34) and (38) from t_{k+1} to t_k, we conclude that

$$\boldsymbol{\gamma}_{k,i} = -N\frac{\partial\mathcal{J}_1}{\partial v_{k,i}}, \quad \forall k,i, \quad \text{where } \mathcal{J}_1 = \frac{1}{N}\sum_{i=1}^{N} r(v_{F,i}). \tag{40}$$

5.3. Connections between the continuous and DSMC adjoint variables

Given an empirical distribution and its limit, we know by the strong law of large numbers that

$$\frac{1}{N}\sum_{i=1}^{N} I_\Omega(v_{F,i}) \xrightarrow{a.s.} \int_\Omega f(v,T)dv$$

for every measurable set $\Omega \subseteq \mathbb{R}^3$, where I_Ω is the indicator function. Also,

$$\lim_{N\to\infty} \mathcal{J}_1 = \lim_{N\to\infty} \frac{1}{N}\sum_{i=1}^{N} r(v_{F,i}) = \int_{\mathbb{R}^3} r(v) f(v,T)dv = J_1.$$

Next, we derive the connections between the two derivatives at any time step t_k,

$$\frac{\delta J_1}{\delta f(v,t_k)} \quad \text{and} \quad \frac{\partial \mathcal{J}_1}{\partial v_{k,i}},$$

which will directly uncover the relationship between the continuous and DSMC adjoint variables based on (37) and (40).

We consider two empirical density functions,

$$\frac{1}{N}\sum_{i=1}^{N}\delta(v - v_{k,i}) \quad \text{and} \quad \frac{1}{N}\sum_{i=1}^{N}\delta(v - \bar{v}_{k,i}), \quad \bar{v}_{k,i} = v_{k,i} + \delta v_{k,i},$$

as the approximations to the density functions $f(v,t_k)$ and $f(v,t_k)+\delta f(v,t_k)$, respectively. We remark that $v_{k,i}$ are random variables following the distribution function $f(v,t_k)$, while we choose $\delta v_{k,i}$ to be deterministic, as follows:

$$\delta v_{k,i} = \begin{cases} \eta, & v_{k,i} \in \Omega, \\ 0, & v_{k,i} \in \mathbb{R}^3 \backslash \Omega. \end{cases}$$

Here, Ω is an arbitrary measurable set and η is a small constant vector.

The first-order variation of the continuous and the discrete objective functions can be stated as below. For any time step t_k,

$$\delta J_1 = \int_{\mathbb{R}^3} \frac{\delta J_1}{\delta f}\delta f dv = \int_{\mathbb{R}^3} \frac{\delta J_1}{\delta f(v,t_k)}\Big(f(v,t_k) + \delta f(v,t_k) - f(v,t_k)\Big)dv, \tag{41}$$

$$\delta \mathcal{J}_1 = \sum_{i=1}^{N} \delta v_{k,i} \cdot \frac{\partial \mathcal{J}_1}{\partial v_{k,i}}. \tag{42}$$

Consider $v_i = v_{k,i}$ as one of the N random variables following the distribution $f(v,t_k)$, and denote $\phi(v) = \frac{\delta J_1}{\delta f(v,t_k)}$. Based on (41), we have

$$\begin{aligned}
\delta J_1 &= \int_{\mathbb{R}^3} \frac{\delta J_1}{\delta f(v,t_k)}\big(f(v,t_k) + \delta f(v,t_k)\big)dv - \int_{\mathbb{R}^3} \frac{\delta J_1}{\delta f(v,t_k)} f(v,t_k)dv, \\
&= \mathbb{E}_{\bar{v}_i}\left[\frac{\delta J_1}{\delta f(\bar{v}_i,t_k)}\right] - \mathbb{E}_{v_i}\left[\frac{\delta J_1}{\delta f(v_i,t_k)}\right] \\
&= \mathbb{E}_{\bar{v}_i}[\phi(\bar{v}_i)] - \mathbb{E}_{v_i}[\phi(v_i)] \\
&= \mathbb{E}_{v_i}\big[I_\Omega(v_i)\phi(v_i+\eta) + I_{\mathbb{R}^3\backslash\Omega}(v_i)\phi(v_i)\big] - \mathbb{E}_{v_i}[\phi(v_i)] \\
&= \mathbb{E}_{v_i}[I_\Omega(v_i)\phi(v_i+\eta)] - \mathbb{E}_{v_i}[I_\Omega(v_i)\phi(v_i)], \\
&= \mathbb{E}_{v_i}[I_\Omega(v_i)\left(\phi(v_i+\eta) - \phi(v_i)\right)] \\
&\approx \eta \cdot \mathbb{E}_{v_i}\big[I_\Omega(v_i)\phi'(v_i)\big]
\end{aligned} \tag{43}$$

On the other hand, (41) is related to (42) as the following

$$\delta J_1 = \mathbb{E}_{v_1,\dots,v_N}[\delta \mathcal{J}_1] = \sum_{i=1}^{N} \mathbb{E}_{v_i}\left[\delta v_i \cdot \frac{\partial \mathcal{J}_1}{\partial v_i}\right] = N\eta \cdot \mathbb{E}_{v_i}\left[I_\Omega(v_i)\frac{\partial \mathcal{J}_1}{\partial v_i}\right]. \tag{44}$$

We assume that all $v_i = v_{k,i}$, for $i = 1, \dots, N$, are (approximately) i.i.d. random variables following the same velocity distribution $f(v, t_k)$. Therefore, by combining (43) and (44), we have

$$\eta \cdot \mathbb{E}_{v_i}\left[I_\Omega(v_i)\phi'(v_i)\right] = N\eta \cdot \mathbb{E}_{v_i}\left[I_\Omega(v_i)\frac{\partial \mathcal{J}_1}{\partial v_i}\right].$$

Since η is an arbitrarily small vector in $\mathbb{R}^3$, then

$$\mathbb{E}_v\left[I_\Omega(v)\phi'(v)\right] = N\mathbb{E}_{v_i}\left[I_\Omega(v_i)\frac{\partial \mathcal{J}_1}{\partial v_i}\right] = N\mathbb{E}_v\left[\mathbb{E}_{v_i}\left[I_\Omega(v_i)\frac{\partial \mathcal{J}_1}{\partial v_i}\middle| v_i = v\right]\right] = \mathbb{E}_v\left[I_\Omega(v)\mathbb{E}_{v_i}\left[N\frac{\partial \mathcal{J}_1}{\partial v_i}\middle| v_i = v\right]\right],$$

where we change the variable from v_i to v for the first term and apply the law of total expectation to the second term.

Since the set Ω is arbitrary, $\phi'(v) = (\frac{\delta J_1}{\delta f(v,t_k)})' = -\gamma'(v, t_k)$ and $N\frac{\partial \mathcal{J}_1}{\partial v_{k,i}} = -\boldsymbol{\gamma}_{k,i}$, which have been shown previously in (37) and (40), we obtain the following equation

$$\gamma'(v, t_k) = \mathbb{E}[\boldsymbol{\gamma}_{k,i} | v_{k,i} = v]. \tag{45}$$

Equation (45) serves as a bridge connecting the two adjoint systems that we have derived in Section 3 and Section 4.

5.4. The unbiasedness of the adjoint DSMC gradient estimator

In (20) and (29), we derive two gradient formulae, $\partial_\alpha J$ and $\partial_\alpha \mathcal{J}$, through the continuous and the discrete adjoint systems, respectively. Next, we show the unbiasedness of the gradient estimator $\partial_\alpha \mathcal{J}$ computed through the adjoint DSMC approach:

$$\partial_\alpha J = \mathbb{E}_{v_{0,1},\dots,v_{0,N}\sim f_0(v;\alpha)}[\partial_\alpha \mathcal{J}]. \tag{46}$$

There are a direct way and an indirect way to draw samples from continuous distributions $f(v, 0) = f_0(v; \alpha)$, which is also our initial distribution for the Boltzmann equation:

Direct way: $v_{0,i} \sim f_0(v; \alpha)$

Indirect way: $v_{0,i} = g(\epsilon_{0,i}, \alpha)$, $\epsilon_{0,i} \sim \phi(\epsilon)$.

The second and indirect approach is to first sample from a simpler base distribution $\phi(\epsilon)$, which is independent of the parameter α, and then transform this variate through a deterministic path $g(\epsilon, \alpha)$. This is often referred to as the sampling path or the sampling process [27,37]. For an invertible path, we have the mass-preserving equation:

$$\phi(\epsilon) = f_0\Big(g(\epsilon, \alpha); \alpha\Big) \det\Big(\nabla_\epsilon g(\epsilon, \alpha)\Big). \tag{47}$$

Recall in Section 3 where we derive the OTD framework,

$$\partial_\alpha J = \partial_\alpha J_2 = -\partial_\alpha \int_{\mathbb{R}^3} \gamma(v, 0) f_0(v; \alpha) dv = -\partial_\alpha \mathbb{E}_{v\sim f_0(v;\alpha)}[\gamma(v, 0)].$$

Based on the Law of the Unconscious Statistician (LOTUS),

$$\mathbb{E}_{v\sim f_0(v;\alpha)}[\gamma(v, 0)] = \mathbb{E}_{\epsilon\sim\phi(\epsilon)}[\gamma(g(\epsilon, \alpha), 0)]. \tag{48}$$

Therefore, we have

$$\begin{aligned}
\partial_\alpha J &= -\partial_\alpha \mathbb{E}_{v\sim f_0(v;\alpha)}[\gamma(v, 0)] \\
&= -\partial_\alpha \mathbb{E}_{\epsilon\sim\phi(\epsilon)}\left[\gamma\Big(g(\epsilon, \alpha), 0\Big)\right] = -\partial_\alpha \left(\int \phi(\epsilon)\gamma\Big(g(\epsilon, \alpha), 0\Big)d\epsilon\right) \\
&= -\int \phi(\epsilon)\left(\frac{\partial \gamma(v, 0)}{\partial v}\bigg|_{v=g(\epsilon,\alpha)} \cdot \frac{\partial g(\epsilon, \alpha)}{\partial \alpha}\right) d\epsilon \\
&= -\int f_0\Big(g(\epsilon, \alpha); \alpha\Big)\left(\frac{\partial \gamma(v, 0)}{\partial v}\bigg|_{v=g(\epsilon,\alpha)} \cdot \frac{\partial g(\epsilon, \alpha)}{\partial \alpha}\right) \det\Big(\nabla_\epsilon g(\epsilon, \alpha)\Big)d\epsilon \quad \text{(by Equation (47))} \\
&= -\int f_0(v; \alpha)\left(\frac{\partial \gamma(v, 0)}{\partial v} \cdot \frac{\partial v}{\partial \alpha}\right) dv.
\end{aligned}$$

The last equality is based on the change of variable $v = g(\epsilon, \alpha)$. Together with Equation (45), we have

$$\partial_\alpha J = -\int f_0(v;\alpha)\left(\frac{\partial \gamma(v,0)}{\partial v}\cdot\frac{\partial v}{\partial \alpha}\right)dv = -\frac{1}{N}\sum_{i=1}^{N}\mathbb{E}_{v_{0,i}\sim f_0(v;\alpha)}\left[\boldsymbol{\gamma}_{0,i}\cdot\frac{\partial v_{0,i}}{\partial \alpha}\right]$$

$$= \mathbb{E}_{v_{0,1},\dots,v_{0,N}\sim f_0(v;\alpha)}\left[-\frac{1}{N}\sum_{i=1}^{N}\boldsymbol{\gamma}_{0,i}\cdot\frac{\partial v_{0,i}}{\partial \alpha}\right] = \mathbb{E}_{v_{0,1},\dots,v_{0,N}\sim f_0(v;\alpha)}\left[\partial_\alpha \mathcal{J}\right].$$

In conclusion, $\partial_\alpha \mathcal{J}$ is an unbiased pathwise Monte Carlo gradient estimator for the objective function J.

If we solve the continuous adjoint equation (19a) numerically and then compute the gradient for α following (20), the result should match the gradient computed by the DSMC adjoint approach in Section 4 within numerical error. For example, (19a) can be solved numerically by the DSMC-type scheme which we propose in Section 3.2. The continuous and the DSMC adjoint systems are equivalent in optimizing the model parameter α. We will present several numerical examples that verify the gradient accuracy and demonstrate the optimization process in Section 6.

6. Numerical results

In this section we discuss the results of numerical simulations computing the gradients of the objective function (28), $\mathcal{J}_1 = \frac{1}{N}\sum_{i=1}^{N} r(v_{F,i}) \approx \int_{\mathbb{R}^3} r(v) f(v,T)dv = J_1$, at the final time $t = T$ with respect to the parameter α in the initial conditions $f_0(v;\alpha)$. Four different methods of the gradient computation are used here: (i) finite difference method using several forward DSMC simulations with different parameter values, (ii) the adjoint DSMC method, (iii) the DSMC-type scheme for the adjoint equation, (iv) the direct discretization of the continuous adjoint equation (19a). All four methods lead to the same gradient values, but the adjoint DSMC method is the best in terms of performance given we do not need many digits of accuracy.

Here we consider Maxwellian gas, so the distribution function $f(v,t)$ obeys the Boltzmann equation (13) with a collision operator kernel $q(v - v_1, \sigma) = q(\sigma)$. We further assume that $q(\sigma) = 1/(4\pi)$ and $\rho(t) = \int_{\mathbb{R}^3} f(v,t)dv = 1$ (conserved throughout the evolution of the distribution function), and thus $\mu = \rho\int_{S^2} q(\sigma)d\sigma = 1$ in (14).

For the function $r(v)$ in Section 6.1-6.4, we use v_l^2 and v_l^4, $l \in \{x, y, z\}$, so the objective functions are

$$m2_l(t_k) \triangleq T_l(t_k) = \frac{1}{N}\sum_{i=1}^{N}(v_{k,i}^l)^2 \approx \int_{\mathbb{R}^3} v_l^2 f(v,t_k)dv, \quad l \in \{x,y,z\},$$

$$m4_l(t_k) \triangleq \frac{1}{N}\sum_{i=1}^{N}(v_{k,i}^l)^4 \approx \int_{\mathbb{R}^3} v_l^4 f(v,t_k)dv, \quad l \in \{x,y,z\}.$$

They are the second-order and the fourth-order velocity moments of the distribution function in the l-direction at the time $t = t_k$. We have six objective functions in total. For the parameter α we use temperature values in the initial distribution function $\alpha = [T_x^0, T_y^0, T_z^0]$. We further refer to these gradients as $\frac{\partial T_l}{\partial T_p^0}$ and $\frac{\partial m4_l}{\partial T_p^0}$ respectively, $l, p \in \{x, y, z\}$. Here the dimension of α is only 3. Still, a real advantage of these methods is, of course, when the vector α is high dimensional since the described methods allow one to compute all the components of the gradient $\frac{\delta \mathcal{J}_1}{\delta \alpha}$ by doing the only *one* forward DSMC simulation and *one* backward adjoint DSMC simulation.

For all the methods we use the same initial condition, anisotropic Gaussian,

$$f_0(v) = \frac{1}{(2\pi)^{3/2}\sqrt{T_x^0 T_y^0 T_z^0}}\exp\left(-\frac{v_x^2}{2T_x^0} - \frac{v_y^2}{2T_y^0} - \frac{v_z^2}{2T_z^0}\right), \tag{49}$$

where $T_x^0 = 0.5$, $T_y^0 = 1$, $T_z^0 = 1$, and the total density $\rho = 1$. In this case, the solution to the Boltzmann equation (13) will relax to an isotropic Gaussian with the temperature $T_M = (T_x^0 + T_y^0 + T_z^0)/3 = 0.8333(3)$ over time. In all the tests below, we use the forward Euler time-integration scheme with a time-step $\Delta t = 0.1$. The gradients, $\frac{\partial T_l}{\partial T_p^0}$ and $\frac{\partial m4_l}{\partial T_p^0}$, are computed at the final time $t = T = 2$.

6.1. Forward DSMC simulation

We solve the Boltzmann equation (13) with the Nanbu–Babovsky method as described in Algorithm 1. We represent the distribution function with N particles, ranging from 10^6 to 10^8, and sample the initial condition from the distribution in (49). Collisions are performed according to the collision rules (2) and the collision angles are sampled uniformly over a unit sphere according to the collision kernel of Maxwellian particles, $q(v - v_1, \sigma) = 1/(4\pi)$. Based on Algorithm 1, the fraction of particles that collide at every time step is $N_c/N = \Delta t\mu$ and is equal to 10% for $\Delta t = 0.1$, $\mu = 1$. The total kinetic energy $K(t) = T_x(t) + T_y(t) + T_z(t)$ and the total momentum $p(t) = (p_x(t), p_y(t), p_z(t))$, where $p_l(t) = \int_{\mathbb{R}^3} v_l f(v,t)dv$, are

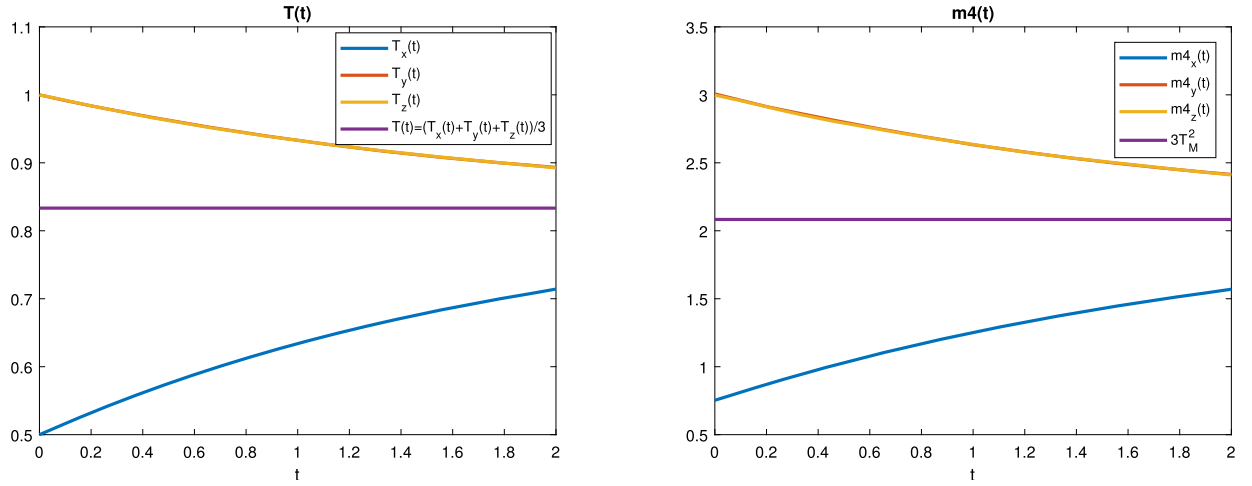

Fig. 4. The relaxation of temperatures $T_x(t), T_y(t), T_z(t)$ to the value $T_M = (T_x^0 + T_y^0 + T_z^0)/3 = 0.8333(3)$ as well as the average temperature $(T_x(t) + T_y(t) + T_z(t))/3$ that stays constant throughout the simulation (left) and the relaxation of fourth moments $m4_x(t), m4_y(t), m4_z(t)$ to the value $3T_M^2 = 2.08333(3)$ from DSMC simulation with initial condition (49), $N = 10^6$ particles and $\Delta t = 0.1$. The red and yellow lines on both graphs practically coincide. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

Table 1
Mean values of $p_x, p_y, p_z, T_x, T_y, T_z$ and $m4_x, m4_y, m4_z$ and their standard deviations at $t = 2$, which are computed using M_s standard DSMC simulations with N particles in each. The temperatures in the initial condition (49) are set to be $[T_x^0\ T_y^0\ T_z^0] = [0.5\ 1\ 1]$.

	$M_s = 100$			
	$N = 10^6$		$N = 10^8$	
	$\bar{X}$	σ_X	$\bar{X}$	σ_X
$p_x(t=2)$	3.5237e-18	1.7446e-17	2.6791e-19	1.9801e-17
$p_y(t=2)$	8.3766e-19	2.4584e-17	1.4893e-18	2.9098e-17
$p_z(t=2)$	-3.5049e-18	2.6808e-17	4.8894e-18	2.8732e-17
$T_x(t=2)$	0.71405	0.00077	0.7138388	0.0000735
$T_y(t=2)$	0.89289	0.00087	0.8930843	0.0000890
$T_z(t=2)$	0.89306	0.00105	0.8930770	0.0001009
$m4_x(t=2)$	1.5699	0.0043	1.5690043	0.0004608
$m4_y(t=2)$	2.4097	0.0059	2.4114132	0.0005268
$m4_z(t=2)$	2.4115	0.0070	2.4113140	0.0006946

conserved by construction of the algorithm, since every pair-wise collision is elastic. Fig. 4 shows the relaxation of temperatures $T_x(t), T_y(t), T_z(t)$ towards the value T_M as well as the relaxation of the fourth-order moments $m4_x(t), m4_y(t), m4_z(t)$ towards the value $3T_M^2$ as functions of time.

Running the standard DSMC simulations several times with randomly sampled initial conditions allows us to estimate the mean values and the standard deviations of quantities of interest at $t = T = 2$. Details are shown in Table 1, where the standard deviation σ_X is computed using $M_s = 100$ simulations with $N = 10^6, 10^8$ particles in each. The standard deviations of the quantities of interest give us estimates of the random errors in DSMC algorithm with N particles. They scale with $1/\sqrt{N}$ which can be seen from the σ_X values in Table 1. The actual standard deviations of the mean values $\bar{X}$ in Table 1 computed using M_s independent simulations are approximately $1/\sqrt{M_s}$ times the values of σ_X for one simulation from Table 1. That is, $\sigma_{\bar{X}} = \sigma_X/\sqrt{M_s}$. We further estimate random errors in the mean values $\bar{X}$ by using 95% of the trust interval of radius $2\sigma_{\bar{X}}$, so $e_{\bar{X}}^{rand} = 2\sigma_{\bar{X}} = \frac{2\sigma_X}{\sqrt{M_s}}$. Finally, we have an estimate of expectations:

$$\mathbb{E}[X] \approx \bar{X} \pm \frac{2\sigma_X}{\sqrt{M_s}}. \tag{50}$$

Based on this approach and the values of σ_X from Table 1, we can see that when $N = 10^8$ and $M_s = 100$, the random errors of $T_x(t=2), T_y(t=2), T_z(t=2)$ are $e_{T_l}^{rand} = 2\sigma_{T_l}/\sqrt{M_s} \approx 0.00002$ (or relative error $e_{T_l}^{rand}/T_l \approx 0.002\%$). The random errors of $m4_x(t=2), m4_y(t=2), m4_z(t=2)$ are $e_{m4_l}^{rand} = 2\sigma_{m4_l}/\sqrt{M_s} \approx 0.0001$ (or relative error $e_{m4_l}^{rand}/m4_l \approx 0.005\%$), $l \in \{x, y, z\}$.

To directly compute the gradients of the velocity moments with respect to the parameter $\alpha = [T_x^0, T_y^0, T_z^0]$ by finite difference, we need to perturb the parameter by a small amount and then compute the corresponding target values at those different initial temperatures. For each component in $[T_x^0\ T_y^0\ T_z^0]$, we apply a perturbation of size $\Delta T_l^0 = 0.1$ or 0.05

Table 2
Mean values of T_x, T_y, T_z and $m4_x, m4_y, m4_z$ at $t = 2$ which are computed using $M_s = 100$ standard DSMC simulations with $N = 10^8$ particles in each. The initial condition (49) is set with the given initial temperatures $[T_x^0\ T_y^0\ T_z^0]$.

$[T_x^0\ T_y^0\ T_z^0]$	[0.5 1 1]	[0.4 1 1]	[0.6 1 1]	[0.5 0.9 1]	[0.5 1.1 1]	[0.5 1 0.9]	[0.5 1 1.1]
$T_x(t=2)$	0.71384	0.65661	0.77108	0.69246	0.73523	0.69246	0.73523
$T_y(t=2)$	0.89308	0.87171	0.91447	0.83586	0.95032	0.87171	0.91447
$T_z(t=2)$	0.89308	0.87169	0.91445	0.87169	0.91445	0.83584	0.95030
$m4_x(t=2)$	1.56900	1.35147	1.80946	1.47149	1.67080	1.47149	1.67080
$m4_y(t=2)$	2.41141	2.30639	2.52069	2.10811	2.73762	2.29959	2.52749
$m4_z(t=2)$	2.41131	2.30628	2.52058	2.29949	2.52737	2.10801	2.73750
$[T_x^0\ T_y^0\ T_z^0]$		[0.45 1 1]	[0.55 1 1]	[0.5 0.95 1]	[0.5 1.05 1]	[0.5 1 0.95]	[0.5 1 1.05]
$T_x(t=2)$		0.68523	0.74247	0.70315	0.72453	0.70315	0.72453
$T_y(t=2)$		0.88240	0.90376	0.86447	0.92171	0.88240	0.90378
$T_z(t=2)$		0.88238	0.90378	0.88238	0.90376	0.86445	0.92168
$m4_x(t=2)$		1.45739	1.68646	1.51973	1.61938	1.51973	1.61938
$m4_y(t=2)$		2.35837	2.46534	2.25690	2.57166	2.35497	2.46893
$m4_z(t=2)$		2.35827	2.46547	2.35487	2.46881	2.25680	2.57155

away from the original values of $[T_x^0\ T_y^0\ T_z^0] = [0.5\ 1\ 1]$. In Table 2, we gather the resulting mean values of the moments that are computed using $M_s = 100$ simulations and $N = 10^8$ particles in each simulation.

Using the values in Table 2, we can estimate the gradients $\frac{\partial \mathcal{J}_1}{\partial \alpha}(\alpha_0)$ via the central finite difference

$$\frac{\partial \mathcal{J}_1}{\partial \alpha}(\alpha_0) \approx \frac{\mathcal{J}_1(\alpha_0 + \Delta\alpha) - \mathcal{J}_1(\alpha_0 - \Delta\alpha)}{2\Delta\alpha}. \tag{51}$$

We can also estimate their errors $e_{\frac{\partial \mathcal{J}_1}{\partial \alpha}(\alpha_0)}$, as shown in Table 3. The error in $\frac{\partial \mathcal{J}_1}{\partial \alpha}(\alpha_0)$ consists of two parts:

$$e_{\frac{\partial \mathcal{J}_1}{\partial \alpha}(\alpha_0)} = e^{FD}_{\frac{\partial \mathcal{J}_1}{\partial \alpha}(\alpha_0)} + e^{rand}_{\frac{\partial \mathcal{J}_1}{\partial \alpha}(\alpha_0)}, \tag{52}$$

where the finite difference error can be estimated as

$$e^{FD}_{\frac{\partial \mathcal{J}_1}{\partial \alpha}(\alpha_0)} \approx \frac{\partial^3 \mathcal{J}_1}{\partial \alpha^3}(\alpha_0)\frac{(\Delta\alpha)^2}{6} \approx \frac{-\mathcal{J}_1(\alpha_0 - 2\widetilde{\Delta\alpha}) + 2\mathcal{J}_1(\alpha_0 - \widetilde{\Delta\alpha}) - 2\mathcal{J}_1(\alpha_0 + \widetilde{\Delta\alpha}) + \mathcal{J}_1(\alpha_0 + 2\widetilde{\Delta\alpha})}{12(\widetilde{\Delta\alpha})^3}(\Delta\alpha)^2, \tag{53}$$

where $\widetilde{\Delta\alpha}$ is a finite step used to estimate $\frac{\partial^3 \mathcal{J}_1}{\partial \alpha^3}$, and the random error can be estimated roughly as

$$e^{rand}_{\frac{\partial \mathcal{J}_1}{\partial \alpha}(\alpha_0)} \approx \frac{e^{rand}_{\mathcal{J}_1(\alpha_0+\Delta\alpha)} + e^{rand}_{\mathcal{J}_1(\alpha_0-\Delta\alpha)}}{2\Delta\alpha} \approx \frac{e^{rand}_{\mathcal{J}_1(\alpha_0)}}{\Delta\alpha}. \tag{54}$$

Notice that the finite difference error is proportional to $(\Delta\alpha)^2$ while the random error is proportional to $1/\Delta\alpha$, meaning that there is an optimal value of

$$\Delta\alpha^* = \left(\frac{3e^{rand}_{\mathcal{J}_1(\alpha_0)}}{\frac{\partial^3 \mathcal{J}_1}{\partial \alpha^3}(\alpha_0)}\right)^{1/3} \tag{55}$$

for a given value of the random error $e^{rand}_{\mathcal{J}_1(\alpha_0)}$ in $J(\alpha_0)$ and the third-order derivative of $\mathcal{J}_1$ at α_0 (unknown a priori) for which the total error is minimal. We compute the third-order derivatives according to (53) with $\widetilde{\Delta\alpha} = 0.05$. Together with the previously computed $e^{rand}_{T_l}$ and $e^{rand}_{m4_l}$, we estimate the optimal step size $\Delta\alpha^* \in (0.06, 1.5)$, for various components of $\frac{\partial T_l}{\partial T_p^0}$ and $\frac{\partial m4_l}{\partial T_p^0}$. Eventually, we opt for $\Delta\alpha = \Delta T_p^0 = 0.1$ for the computation of the gradients and the estimations of the errors that are shown in Table 3. The random errors reflected in the last column of Table 3 are computed using the standard deviations σ_{T_l} and σ_{m4_l} from the last column of Table 1 together with (50) and (54). Comparing the random errors in the last column with the finite-difference errors in the other columns of Table 3, we observe that our choice of $\Delta\alpha = 0.1$ is nearly optimal for $p = x$ components and smaller than optimal for the $p = y, z$ components, but the current value of $\Delta\alpha$ is good enough for our purposes.

6.2. Adjoint DSMC

In the adjoint DSMC method for the gradient calculation, we first solve the Boltzmann equation (13) as in the previous examples, but at each time step $t = t_k$ the indices of the particles that collided, their collision angles $\sigma_{k,i}$ and the pre-collision relative unit velocities $(v_{k,i} - \tilde{v}_{k,i})^\wedge$ are stored in memory for the later use in the backward solve.

Table 3
Gradients $\frac{\delta \mathcal{J}_1}{\delta \alpha}$ that are computed using the finite difference formula (51), estimates of the corresponding finite-difference errors based on (53) and Table 2 values, and the corresponding random errors that are computed based on (54) and Table 1 values. The objective functions $\mathcal{J}_1 = \{T_l(t=2), m4_l(t=2)\}$ and parameter $\alpha = T_p^0$, $l, p \in \{x, y, z\}$. The random errors are the same for all $\alpha = T_p^0$, $p \in \{x, y, z\}$. All quantities here are computed using $\Delta\alpha = 0.1$.

	$\frac{\partial \mathcal{J}_1}{\partial T_x^0} \pm e^{FD}_{\frac{\partial \mathcal{J}_1}{\partial T_x^0}}$	$\frac{\partial \mathcal{J}_1}{\partial T_y^0} \pm e^{FD}_{\frac{\partial \mathcal{J}_1}{\partial T_y^0}}$	$\frac{\partial \mathcal{J}_1}{\partial T_z^0} \pm e^{FD}_{\frac{\partial \mathcal{J}_1}{\partial T_z^0}}$	$e^{rand}_{\frac{\partial \mathcal{J}_1}{\partial T_p^0}}$
$\mathcal{J}_1 = T_x(t=2)$	0.572335 ± 0.0001	0.213839 ± 2.1e-06	0.213836 ± 8.6e-07	± 0.0002
$\mathcal{J}_1 = T_y(t=2)$	0.213831 ± 0.0003	0.572331 ± 2.1e-06	0.213842 ± 3.1e-07	± 0.0002
$\mathcal{J}_1 = T_z(t=2)$	0.213834 ± 0.0002	0.213830 ± 2.8e-08	0.572322 ± 5.6e-07	± 0.0002
$\mathcal{J}_1 = m4_x(t=2)$	2.289942 ± 0.0011	0.996551 ± 7.5e-06	0.996538 ± 6.0e-06	± 0.0009
$\mathcal{J}_1 = m4_y(t=2)$	1.071492 ± 0.0025	3.147580 ± 7.5e-06	1.139506 ± 5.5e-06	± 0.0011
$\mathcal{J}_1 = m4_z(t=2)$	1.071524 ± 0.0006	1.139418 ± 1.1e-06	3.147446 ± 5.7e-07	± 0.0014

Table 4
Gradients $\frac{\delta \mathcal{J}_1}{\delta \alpha}$ computed by the adjoint DSMC approach and estimations of the corresponding random errors based on formula (50), where the objective function $\mathcal{J}_1 = \{T_l(t=2), m4_l(t=2)\}$ and the parameter $\alpha = T_p^0$, $l, p \in \{x, y, z\}$.

	$\frac{\partial \mathcal{J}_1}{\partial T_x^0} \pm e^{rand}_{\frac{\partial \mathcal{J}_1}{\partial T_x^0}}$	$\frac{\partial \mathcal{J}_1}{\partial T_y^0} \pm e^{rand}_{\frac{\partial \mathcal{J}_1}{\partial T_y^0}}$	$\frac{\partial \mathcal{J}_1}{\partial T_z^0} \pm e^{rand}_{\frac{\partial \mathcal{J}_1}{\partial T_z^0}}$
$\mathcal{J}_1 = T_x(t=2)$	0.572316 ± 1.1e-05	0.213836 ± 8.4e-06	0.213835 ± 7.7e-06
$\mathcal{J}_1 = T_y(t=2)$	0.213846 ± 9.5e-06	0.572337 ± 1.1e-05	0.213839 ± 9.5e-06
$\mathcal{J}_1 = T_z(t=2)$	0.213838 ± 8.9e-06	0.213828 ± 7.5e-06	0.572325 ± 1.2e-05
$\mathcal{J}_1 = m4_x(t=2)$	2.289879 ± 1.3e-04	0.996589 ± 8.6e-05	0.996541 ± 8.3e-05
$\mathcal{J}_1 = m4_y(t=2)$	1.071577 ± 9.1e-05	3.147648 ± 1.8e-04	1.139492 ± 9.0e-05
$\mathcal{J}_1 = m4_z(t=2)$	1.071486 ± 8.0e-05	1.139424 ± 8.0e-05	3.147454 ± 1.7e-04

Afterwards, we solve the adjoint DSMC equations derived in Section 4 backward in time. Similar to the definition of the forward particles (15), at the k-th time interval, we represent the adjoint particles as

$$\mathbf{\Gamma}_k = \{\boldsymbol{\gamma}_1, \dots, \boldsymbol{\gamma}_i, \dots, \boldsymbol{\gamma}_N\}(t_k),$$

and we denote the i-th adjoint particle in $\mathbf{\Gamma}_k$ as $\boldsymbol{\gamma}_{k,i}$. We remark that $\boldsymbol{\gamma}_{k,i}$ is a vector in $\mathbb{R}^3$. We initialize our backward solve with the "final conditions" (30), $\boldsymbol{\gamma}_{F,i} = -\partial_v r(v_{F,i})$, $i \in 1 \dots N$, at the final time $t = T = 2$, and $r(v)$ is set to be v_l^2 or v_l^4, $l \in \{x, y, z\}$, as discussed previously, where l is fixed for a given simulation. Thus, $\boldsymbol{\gamma}^j_{F,i} = -2v^l_{F,i}\delta_{l,j}$ and $\boldsymbol{\gamma}^j_{F,i} = -4(v^l_{F,i})^3\delta_{l,j}$, respectively, where $l, j \in \{x, y, z\}$ and $\delta_{l,j} = 1$ if $l = j$ and 0 otherwise.

The adjoint DSMC equations (33a)-(33b) are then solved backward in time for each collision that happened during the forward solve using the collision angles and the relative unit velocities that have been stored during the forward solve. Once the backward time evolution reaches the initial time $t = 0$, we use (29), $\frac{\partial \mathcal{J}_1}{\partial \alpha} = -\frac{1}{N}\sum_{i=1}^N \boldsymbol{\gamma}_{I,i} \cdot \partial_\alpha v_{0,i}(\alpha)$, to calculate the gradient of the objective function with respect to the model parameter α. Here, $\partial_\alpha v_{0,i}(\alpha)$ is a derivative of the initial sample of particles with respect to the parameter α. Hence, it is important to have not only an initial distribution function $f_0(v, \alpha)$ that depends continuously on the parameter α but also a particular way of sampling such that *the samples depend continuously on the parameter* α. Since our initial distribution is an isotropic Gaussian (49), we can sample it by sampling $3N$ values from the standard normal distribution $\mathcal{N}(0, 1)$ and then rescaling the values with appropriate initial temperatures as

$$v_{0,i} = (v^x_{0,i}, v^y_{0,i}, v^z_{0,i}) = (\sqrt{T_x^0}v^{x\mathcal{N}}_{0,i}, \sqrt{T_y^0}v^{y\mathcal{N}}_{0,i}, \sqrt{T_z^0}v^{z\mathcal{N}}_{0,i}),$$

where $v^{x\mathcal{N}}_{0,i}, v^{y\mathcal{N}}_{0,i}, v^{z\mathcal{N}}_{0,i}$ are samples of $\mathcal{N}(0, 1)$. For $\alpha = T_p^0$, we can easily compute

$$\frac{\partial v^j_{0,i}}{\partial T_p^0} = \frac{v^{j\mathcal{N}}_{0,i}}{2\sqrt{T_p^0}}\delta_{j,p} = \frac{v^j_{0,i}}{2T_p^0}\delta_{j,p}, \quad j, p \in \{x, y, z\}.$$

We perform $M_s = 100$ forward DSMC and backward adjoint DSMC simulations with $N = 10^8$ particles for each simulation. Based on (50), we compute the mean values and random errors for quantities $\frac{\partial T_l}{\partial T_p^0}$ and $\frac{\partial m4_l}{\partial T_p^0}$, $l, p \in \{x, y, z\}$. The results are gathered in Table 4, where the values of the gradients match those in Table 3 up to 4-5 significant digits. The random errors in Table 4 are approximately 10^{-5} for $\frac{\partial T_l}{\partial T_p^0}(t=2)$ and 10^{-4} for $\frac{\partial m4_l}{\partial T_p^0}(t=2)$, which is about one order of magnitude smaller than the ones in Table 3. It demonstrates another advantage of this method that it does not suffer from amplified random errors compared to the finite difference calculations. For the latter, the random error increases while computing the gradients due to the subtraction of two close values in (51).

Table 5
Gradients $\frac{\delta \mathcal{J}_1}{\delta \alpha}$ computed by the DSMC-type scheme for the adjoint equation and estimations of the corresponding random errors based on formula (50), where the objective function $\mathcal{J}_1 = \{T_l(t=2), m4_l(t=2)\}$ and the parameter $\alpha = T_p^0$, $l, p \in \{x, y, z\}$.

	$\frac{\partial \mathcal{J}_1}{\partial T_x^0} \pm e^{rand} \frac{\partial \mathcal{J}_1}{\partial T_x^0}$	$\frac{\partial \mathcal{J}_1}{\partial T_y^0} \pm e^{rand} \frac{\partial \mathcal{J}_1}{\partial T_y^0}$	$\frac{\partial \mathcal{J}_1}{\partial T_z^0} \pm e^{rand} \frac{\partial \mathcal{J}_1}{\partial T_z^0}$
$\mathcal{J}_1 = T_x(t=2)$	0.572229 ± 3.6e-03	0.213398 ± 8.9e-04	0.213809 ± 1.1e-03
$\mathcal{J}_1 = T_y(t=2)$	0.211871 ± 4.3e-03	0.573324 ± 2.6e-03	0.213879 ± 1.7e-03
$\mathcal{J}_1 = T_z(t=2)$	0.216833 ± 3.2e-03	0.211929 ± 1.3e-03	0.572561 ± 2.3e-03
$\mathcal{J}_1 = m4_x(t=2)$	2.285860 ± 1.9e-02	0.992066 ± 8.6e-03	0.992431 ± 9.4e-03
$\mathcal{J}_1 = m4_y(t=2)$	1.066416 ± 2.7e-02	3.147584 ± 1.7e-02	1.133434 ± 1.4e-02
$\mathcal{J}_1 = m4_z(t=2)$	1.079131 ± 1.9e-02	1.127308 ± 9.9e-03	3.145782 ± 2.7e-02

6.3. DSMC-type scheme for the adjoint equation

For the DSMC-type scheme, we need to store the same information during the forward modeling as the adjoint DSMC method. We solve the continuous adjoint equation (19a) backward in time using a *DSMC-type scheme* that we have developed in Section 3.2. In this method, we evolve values of the function $\gamma(v,t)$ evaluated at locations $v_{k,i}$, i.e., the DSMC particles from the forward solve. At the k-th time step, the set

$$\Gamma_k = \{\hat{\gamma}_1, \dots, \hat{\gamma}_i, \dots \hat{\gamma}_N\}(t_k)$$

contains all those function values, where $\hat{\gamma}_i(t_k) = \gamma(v_{k,i}, t_k)$ is i-th adjoint "particle" in Γ_k. We remark that $\hat{\gamma}_i(t_k)$ here is a scalar value, $\hat{\gamma}_i(t_k) \in \mathbb{R}$. We initialize our backward solve with the "final conditions" (19b). That is, $\hat{\gamma}_i(t=T) = -r(v_{F,i})$, $i \in 1 \dots N$, where $\{v_{F,i}\}_{i=1}^N$ are the velocity particles from the forward DSMC at the final time $t = T = 2$. Then the continuous adjoint equation (19a) can be solved backward in time following Algorithm 2, where $\hat{\gamma}_i(t_k)$ are updated at $k+1 \to k$ time step according to (26) only if $v_{k,i}$ particles participated in a collision during the forward solve at the step $k \to k+1$. To approximate the last terms in (26), we perform a linear interpolation using the scattered data Γ_{k+1} at locations $v_{k+1,i}$; see (27). Eventually, once the backward time evolution reaches the initial time $t = 0$, the gradient of the objective function (20) can be computed using the values of $\hat{\gamma}_i(t=0)$, i.e., Γ_0. In case of the initial condition (49) and the parameter $\alpha = T_p^0$, $p \in \{x, y, z\}$, the derivative $\partial_\alpha f_0(v;\alpha)$ of the initial distribution function with respect to the parameter α in (20) becomes

$$\frac{\partial f_0(v;\alpha)}{\partial \alpha} = \left(\frac{v_p^2}{T_p^0} - 1\right) \frac{1}{2T_p^0} f_0(v). \tag{56}$$

By approximating the initial distribution function with the sample particles $f_0(v) \approx \frac{1}{N}\sum_{i=1}^N \delta(v - v_{0,i})$, we get

$$\begin{aligned} \frac{\partial J_1}{\partial \alpha} &= -\int \gamma(v,0) \frac{\partial f_0(v;\alpha)}{\partial \alpha} dv = -\int \gamma(v,0) \left(\frac{v_p^2}{T_p^0} - 1\right) \frac{1}{2T_p^0} f_0(v) dv \\ &\approx -\frac{1}{N} \sum_{i=1}^N \frac{\hat{\gamma}_i(t=0)}{2T_p^0} \left(\frac{(v_{0,i}^p)^2}{T_p^0} - 1\right) = \frac{\partial \mathcal{J}_1}{\partial \alpha}. \end{aligned} \tag{57}$$

We performed $M_s = 10$ forward DSMC simulations and backward simulations of the adjoint equation using the DSMC-type scheme with $N = 10^6$ particles for each simulation. We choose smaller N and M_s here because this scheme works much slower due to the interpolation procedure at every time step for every colliding particle. We compute the mean values of the gradients according to (57), and estimate their random errors based on (50). The results are presented in Table 5. We observe that the gradients in Table 4 and Table 5 match up to 2-4 significant digits. The random errors in Table 5 are approximately 0.003 for $\frac{\partial T_l}{\partial T_p^0}(t=2)$ and 0.02 for $\frac{\partial m4_l}{\partial T_p^0}(t=2)$, which are about two orders of magnitude larger than the ones in Table 4 as a result of using $N = 10^6$ instead of $N = 10^8$ and the additional error introduced by numerical interpolation.

6.4. Direct discretization of the adjoint equation integrals

Here we solve the continuous equation (19a) for $\gamma(v,t)$ from $t = T = 2$ to $t = 0$ with the "final condition" (19b) using a direct numerical integration scheme to compute the integral term on the right-hand side. See details of the numerical scheme in Appendix A.

As previously, we consider the gradients of the second-order moments, $T_l(t=2)$, and the fourth-order moments, $m4_l(t=2)$, with respect to $\alpha = T_p^0$, $l, p \in \{x, y, z\}$. For each objective function, we perform one forward DSMC simulation with $N =$

Table 6
Gradients $\frac{\delta J_1}{\delta \alpha}$ computed by the direct discretization of the integrals in the continuous adjoint equation (19a), where $J_1 = \{T_l(t=2), m4_l(t=2)\}$, $\alpha = T_p^0$, $l, p \in \{x, y, z\}$.

	$\frac{\partial J_1}{\partial T_x^0}$	$\frac{\partial J_1}{\partial T_y^0}$	$\frac{\partial J_1}{\partial T_z^0}$
$J_1 = T_x(t=2)$	0.573407	0.214738	0.214742
$J_1 = T_y(t=2)$	0.214944	0.572962	0.214757
$J_1 = T_z(t=2)$	0.211709	0.211531	0.569767
$J_1 = m4_x(t=2)$	2.343578	1.027094	1.026818
$J_1 = m4_y(t=2)$	1.099767	3.193492	1.165838
$J_1 = m4_z(t=2)$	1.072768	1.137018	3.153235

Table 7
The comparison of the four methods that compute the gradients of the Boltzmann-constrained optimization problems, in terms of memory requirements, error scaling and operation count.

	memory requirements, bytes	error scaling	operation count
(i) finite difference	$24N$	$\mathcal{O}((\Delta\alpha)^2) + \mathcal{O}(\frac{1}{\sqrt{N\Delta\alpha}})$	$23.5(D_\alpha+1)N\mu T$
(ii) adjoint DSMC method	$24N$ + $(24N + 28N\mu T)$	$\mathcal{O}(\frac{1}{\sqrt{N}})$	$17.5N\mu T$
(iii) DSMC-type scheme	$24N$ + $(8N + 28N\mu T)$	$\mathcal{O}(\frac{1}{\sqrt{N}})$	$\approx (40 + \frac{C}{\Delta t}\log(N))N\mu T$
(iv) direct discretization	$8n_{\text{grid}}^3(\frac{T}{\Delta t}+2)$	$\mathcal{O}(\frac{1}{\sqrt{N}})$ + $\mathcal{O}((\Delta v)^2)$	$\approx (21n_\varphi n_\theta + 4)n_{\text{grid}}^6\frac{T}{\Delta t}$

Table 8
CPU run time of the four different methods presented in Section 6.1-6.4 under different parameters.

	$N = 10^6$	$N = 10^7$	$N = 10^8$	$n_{\text{grid}} = 30$	$n_{\text{grid}} = 40$
forward DSMC simulation (Algorithm 1)	0.38 s	5 s	60 s		
adjoint DSMC simulation (Algorithm 3)	0.22 s	2.7 s	30 s		
the DSMC-type scheme (Algorithm 2)	280 s	4100 s			
direct discretization of the equation (19a)				25000 s	126000 s

10^8 particles and one backward simulation via the scheme described in (A.2). We did only one simulation ($M_s = 1$) for each objective function due to its extensively long computational time; see the next subsection for discussions on performance. The results are gathered in Table 6. Values of the gradients in Table 6 and Table 4 match up to 2-3 significant digits. We still have random errors that are contributed through the values of $f(v,t)$ in the forward DSMC. The random errors in Table 6 can be roughly estimated using two standard deviations in the last column of Table 1, or $e_{T_l}^{rand} = 2\sigma_{T_l}/\sqrt{M_s} \approx 0.0002$ and $e_{m4_l}^{rand} = 2\sigma_{m4_l}/\sqrt{M_s} \approx 0.001$. They are much smaller than the overall errors (dominated by finite discretization errors here) 0.002 for $\frac{\partial T_l(t=2)}{\partial T_p^0}$ and 0.05 for $\frac{\partial m4_l(t=2)}{\partial T_p^0}$, which we can compute by comparing the gradient values in Table 6 and Table 4.

We have also numerically verified Equation (45) using the values of $\gamma(v,t_k)$ at grid points $v = v_{i_x,i_y,i_z}$ obtained during the direct integration simulations and using the finite difference to compute the derivative $\gamma'(v,t_k)$. We approximate $\mathbb{E}[\boldsymbol{\gamma}_{k,i}|v = v_{k,i}]$ using the histogram count of $\boldsymbol{\gamma}_{k,i}$ obtained via the adjoint DSMC method around the same grid points $v = v_{i_x,i_y,i_z}$.

6.5. Method comparison

So far, we have described and demonstrated four different ways to compute the gradient of an objective function numerically after the forward DSMC simulations. The summary of their comparison in terms of memory requirements, error scaling and operation count is given in Table 7. See the details of the comparison in Appendix B.

Table 8 shows the timings of our code we recorded per objective function using Intel Core i7-3770K processor (4 cores @4.5 GHz) and $N = 10^6, 10^7, 10^8$, $T = 2$, $\Delta t = 0.1$, $n_{\text{grid}} = 30, 40$, $n_\varphi = n_\theta = 10$ parameters.

The adjoint DSMC is slightly faster (up to 20-30%) than the forward DSMC for the same number of particles N (timings for computing the vector σ and the unit collision direction are included in the forward DSMC timings above and take about 20% of the overall timings, whereas they would add about 35% to the adjoint DSMC timings if we were to include them there). The adjoint DSMC is more than 1000 times faster than the DSMC-like scheme and much faster than the direct discretization of (19a). At the same time, the errors in the adjoint DSMC are at least one order of magnitude smaller than in other methods due to absence of finite-difference or interpolation errors; see Tables 3–5 and Section 6.4 where we have error estimates (partly numeric and partly analytic). By performing only one adjoint DSMC simulation, we obtain all the gradient components of the objective function. At the same time, if we use the finite difference method, we need to perform at least $D_\alpha + 1$ simulations, where D_α is the dimensionality of α. The benefits of the adjoint DSMC algorithm particularly stand out when solving large-scale optimization problems.

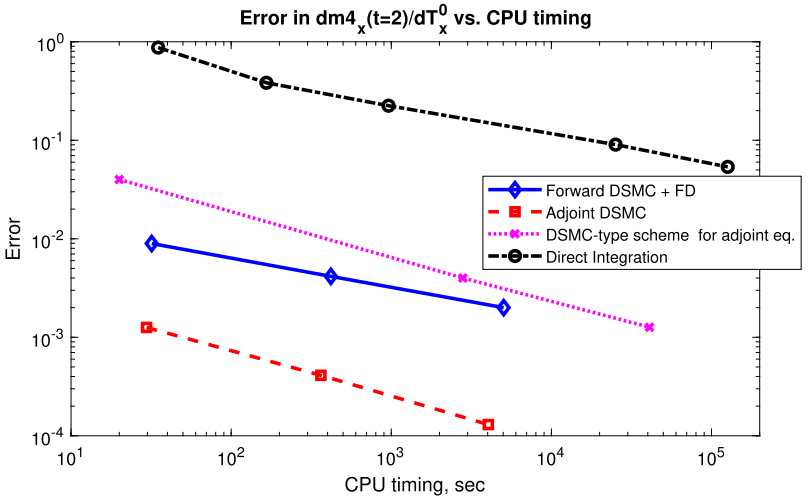

Fig. 5. The error in $\frac{\partial m4_x(t=2)}{\partial T_x^0}$ vs. CPU time for all four methods.

Fig. 5 shows the error in $\frac{\partial m4_x}{\partial T_x^0}$ vs. CPU time measured in simulations for all four methods. Since one forward DSMC solve is needed for all methods, Fig. 5 reflects only CPU timing for the additional computations needed to compute the gradient, namely one extra forward DSMC simulation ($D_\alpha = 1$) for (i) or one backward solution for (ii)-(iv). For (ii), the errors were estimated using formula (50) as the error values in Table 4. For (iii) and (iv) the error values in the figure were computed as a difference between the numerical values $\frac{\partial m4_x}{\partial T_x^0}$ obtained in numerical simulations and the reference value of $\frac{\partial m4_x}{\partial T_x^0}$ from Table 4 as it is the most accurate value we have computed. For (i) the errors were estimated using formulas (52)-(55) and the fact that $e^{rand}_{m4_x(t=2)} \propto 1/\sqrt{N}$ to demonstrate that the error scaling and the actual errors (in comparison to the reference value from Table 4) were $2-10$ times smaller. Note that the slope of the forward DSMC+FD line (blue) is less than the slopes for the adjoint DSMC (red) and DSMC-type scheme for adjoint equation (purple) since the total error in (52) for the optimal $\Delta\alpha^*$ (as in (55)) scales like $\propto 1/N^{1/3}$ and CPU time $\propto N$.

6.6. Optimization examples

We have previously discussed the accuracy and performance of several different numerical schemes to compute the gradient of optimization problems constrained by the Boltzmann equation. The adjoint DSMC method particularly stands out for its simplicity, computational efficiency, and the direct connections with the well-established forward DSMC method [38], as discussed in Section 5.2. Here, we use two optimization examples to illustrate the great potential of the adjoint DSMC method for efficiently solving optimization problems constrained by the Boltzmann equation with the nonlinear collision operator.

6.6.1. Matching the velocity moments

We have been using the velocity moments of the probability distribution at the final time T as the objective function to test the accuracy of the gradients. Here, we follow the earlier discussions and set the first objective function as

$$\mathcal{J}_1(\alpha) = \|\mathbf{d}_1 - \mathbf{d}_2\|_2^2 \tag{58}$$

where $\mathbf{d}_1 = [T_x, T_y, T_z]$, the second velocity moments in each direction and $\mathbf{d}_2 = \frac{1}{2}[m4_x, m4_y, m4_z]$, half of the fourth-order velocity moments in each direction at $t = T = 2$. As we have defined earlier,

$$m4_l(T) = \frac{1}{N}\sum_{i=1}^{N}(v^l_{F,i})^4 \approx \int_{\mathbb{R}^3} v_l^4 f(v,T)dv, \quad T_l(T) = \frac{1}{N}\sum_{i=1}^{N}(v^l_{F,i})^2 \approx \int_{\mathbb{R}^3} v_l^2 f(v,T)dv,$$

for $l \in \{x, y, z\}$. Here, $f(v, T)$ solves the Boltzmann equation (13) given the initial condition (49). We only treat the initial temperature of the y direction, T_y^0, as the unknown parameter α, while fixing $T_x^0 = 0.5$ and $T_z^0 = 1$. This is to avoid the trivial optimal solution $T_x^0 = T_y^0 = T_z^0 = 0$ that minimizes the objective function (58) if $\alpha = [T_x^0, T_y^0, T_z^0]$.

One may notice that the objective function (58) in this example does not match the formulation (18). Nevertheless, it is easy to adapt a general objective function to either the continuous or the DSMC adjoint system based on the role of the adjoint equations: large systems of chain rule which propagate the Fréchet derivative $\frac{\delta J_1}{\delta f(v,T)}$ backward in time to $\frac{\delta J_1}{\delta f(v,0)}$ as

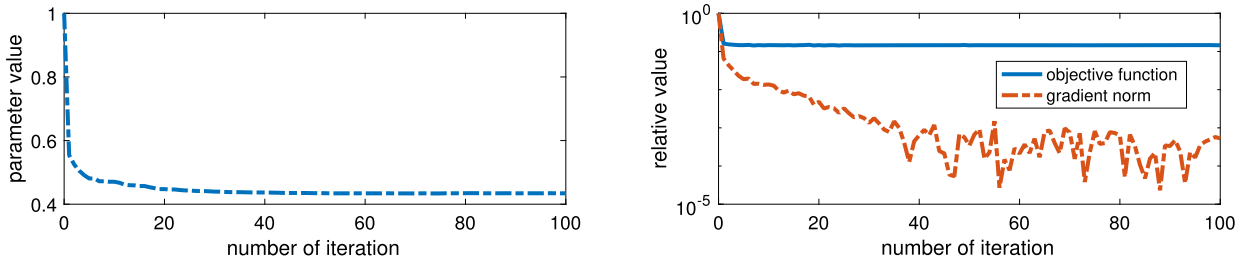

Fig. 6. Left: the convergence history of the parameter T_y^0 for the example discussed in Section 6.6.1. Right: the decrease of the normalized objective function value and the size of the gradient for the first 100 iterations in the optimization step.

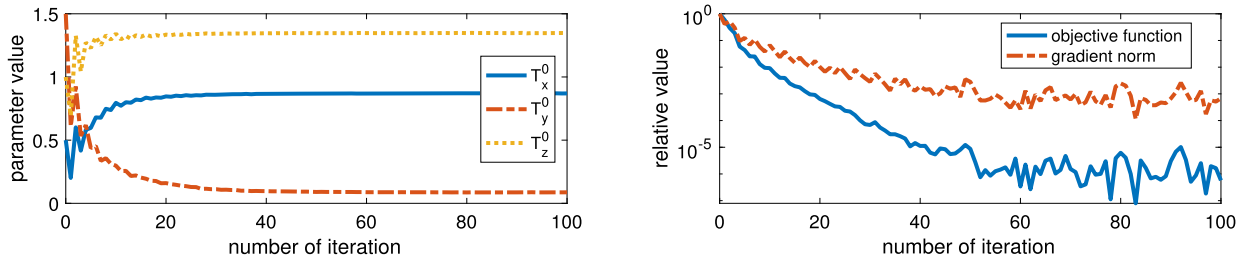

Fig. 7. Left: the convergence history of the three parameters $[T_x^0, T_y^0, T_z^0]$ for the example discussed in Section 6.6.2. Right: the decrease of the normalized objective function value and the ℓ^2 norm of the gradient for the first 100 iterations in the optimization step.

$f(v,0)$ directly depends on the model parameter α. Hence, in this example, the final condition for the continuous adjoint equation (19a) should be

$$\gamma(v,T) = -\frac{\delta J_1}{\delta f(v,T)} = -2(\mathbf{d}_1 - \mathbf{d}_2)\cdot\frac{\delta(\mathbf{d}_1-\mathbf{d}_2)}{\delta f(v,T)} = -2(\mathbf{d}_1-\mathbf{d}_2)\cdot(v^2-\frac{v^4}{2}).$$

The final condition for the DSMC adjoint system, $\boldsymbol{\gamma}_{F,i} = [\boldsymbol{\gamma}^x_{F,i}, \boldsymbol{\gamma}^y_{F,i}, \boldsymbol{\gamma}^z_{F,i}]$, also the adjoint variable for the final particle velocity $v_{F,i} = [v^x_{F,i}, v^y_{F,i}, v^z_{F,i}]$, should be

$$\boldsymbol{\gamma}^l_{F,i} = -\frac{\partial \mathcal{J}_1}{\partial v^l_{F,i}} = -\frac{2}{N}(T_l(T) - m4_l(T))(2v^l_{F,i} - 2(v^l_{F,i})^3),\ l\in x,y,z,\ i=1,2,\ldots,N.$$

Starting with $T_y^0 = 1$ as the initial guess for α, we use a gradient-based optimization algorithm to minimize the objective function (58). The steepest descent method with a backtracking line search following the Armijo–Goldstein condition is applied to find a proper stepsize along the descent direction [39]. We compute the gradient by solving one forward DSMC with the current α, and then one adjoint DSMC is solved in every iteration of the optimization process. The total number of particles in both DSMC simulations is $N = 10^7$. The spacing in the time domain is $\Delta t = 0.1$. The convergence history of this example is shown in Fig. 6. Both the objective function and the size of the gradient monotonically decrease in the first 30 iterations. The convergence slows down as the gradient is smaller than 0.1% of its initial size. We start to observe oscillations in the gradient that come from the random errors in the DSMC solutions. The iterates converge to 0.4344, the global minimum of the objective function if T_y^0 is the only parameter.

6.6.2. Inverse problem

Our second example is based the setup of an inverse problem. The fourth-order velocity moments at the final time $T = 2$ are statistical quantities of interest that can be observed in a realistic or experimental setting. The observable information is solely affected by the unknown initial temperature of the distribution that we aim to recover by minimizing the difference between the observed data and the predicted data simulated by our guess of the model parameter. The reconstruction is formulated as a nonlinear least-squares problem

$$\alpha^* = \arg\min_{\alpha} J_1(\alpha) = \arg\min_{\alpha} \|\mathbf{d}_{\text{obs}} - \mathbf{d}_{\text{pred}}(\alpha)\|_2^2$$

where the predicted data $\mathbf{d}_{\text{pred}}(\alpha) = [m4_x, m4_y, m4_z]$ is a vector of the fourth-order velocity moments at $T = 2$. We set the true data $\mathbf{d}_{\text{obs}} = [2,1,3]$. All other notations and the choice of optimization algorithm follow the previous optimization example.

The initial guess of the parameters is $[0.5, 1.5, 1.0]$. The convergence history of the computational inverse problem is shown in Fig. 7. The three components of α converge to $[0.8670, 0.0870, 1.3470]$ in the first 40 iterations. The same as in the previous example, the small variations in the objective function and the gradient norm for the remaining 60 iterations are introduced by the random errors of DSMC simulations, as seen in the plots. Increasing the number of particles can help mitigate the small perturbations.

Remark 3. We show two simple optimization experiments as examples, but one can apply the framework to more general and large-scale optimization problems constrained by the nonlinear Boltzmann equation. There are at least three directions to generalize the applications. First, the dimensionality of the unknown parameter could be increased with hardly any extra cost. Second, the model parameter is not limited to the initial condition. Such examples include shape optimization of the flow channel [42]. Third, following the same idea, we plan to generalize the adjoint DSMC systems for inhomogeneous Boltzmann equation with the nonlinear collision operator or even more complicated kinetic description.

7. Conclusion

In this paper, we present the OTD and DTO approaches of computing the gradient of optimization problems based on the nonlinear Boltzmann equation. The highlight of both frameworks is that one only needs to solve the Boltzmann equation and the adjoint system once to compute the gradient, independent of the size of the unknown in the optimization. The adjoint DSMC system, derived by the DTO approach, offers a deterministic numerical scheme that is remarkably efficient to implement with the forward DSMC method. On the other hand, the Monte Carlo type method designed for the continuous adjoint equation could potentially be used for the linear Boltzmann equation [11]. We plan to extend both frameworks to the general VHS kernel and to the inhomogeneous case. In particular, the adjoint DSMC approach applies to more general kinetic models for gases and plasmas whose behavior could be modeled by Monte Carlo binary collisions. One of such models is the Coulomb collision for charged particles [47]. As the next step, we will apply the adjoint DSMC methods to realistic optimization problems that occur naturally in a broader class of kinetic applications.

CRediT authorship contribution statement

Russel Caflisch: Conceptualization, Methodology, Writing – original draft, Writing – review & editing. **Denis Silantyev:** Formal analysis, Investigation, Methodology, Software, Validation, Visualization, Writing – original draft, Writing – review & editing. **Yunan Yang:** Formal analysis, Methodology, Writing – original draft, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

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. The authors thank the Courant Institute of Mathematical Sciences, New York University, for research support and computational resources.

Appendix A. Direct numerical integration scheme of the continuous adjoint equation (19b)

We treat $\gamma(v,t)$ as a continuous function with scalar values. We consider a grid in the $v \in \mathbb{R}^3$ space with n_{grid} number of grid points equally spaced in the interval $[-5v_{th}, 5v_{th}]$ for each of the x, y, z directions. We set the thermal velocity $v_{th} = \sqrt{T_M}$, where $T_M = (T_x^0 + T_y^0 + T_z^0)/3$ is the equilibrium temperature. Thus, the grid spacing is $\Delta v = 10v_{th}/(n_{\text{grid}} - 1)$ and the grid points are $v_{i_x,i_y,i_z} = [-5v_{th} + i_x\Delta v, -5v_{th} + i_y\Delta v, -5v_{th} + i_z\Delta v]$, where $i_x, i_y, i_z \in \{0, 1, ..., n_{\text{grid}} - 1\}$.

To propagate equation (19a) backward in time, we need values of $f(v,t)$ at the grid points at each time step. Hence, at each time step of the forward DSMC solve of the Boltzmann equation (13), we compute and store those function values using a 3D histogram with bins of size Δv in each direction that are centered at the grid points v_{i_x,i_y,i_z}.

To solve equation (19a) numerically we first simplify it as follows

$$-\frac{\partial \gamma(v,t)}{\partial t} = \int_{\mathbb{R}^3}\int_{\mathcal{S}^2} (\gamma(v_1') + \gamma(v'))f(v_1)qd\sigma dv_1 - \frac{\mu}{\rho}\int_{\mathbb{R}^3}\gamma(v_1)f(v_1)dv_1 - \mu\gamma(v) \tag{A.1}$$

where v', v_1' are as in (2) and σ is a unit vector spanning the unit sphere. Due to the symmetry $v'(-\sigma) = v_1'(\sigma)$ and the fact that σ spans the whole unit sphere, $\gamma(v_1')$ and $\gamma(v')$ in the first integral give equal contributions.

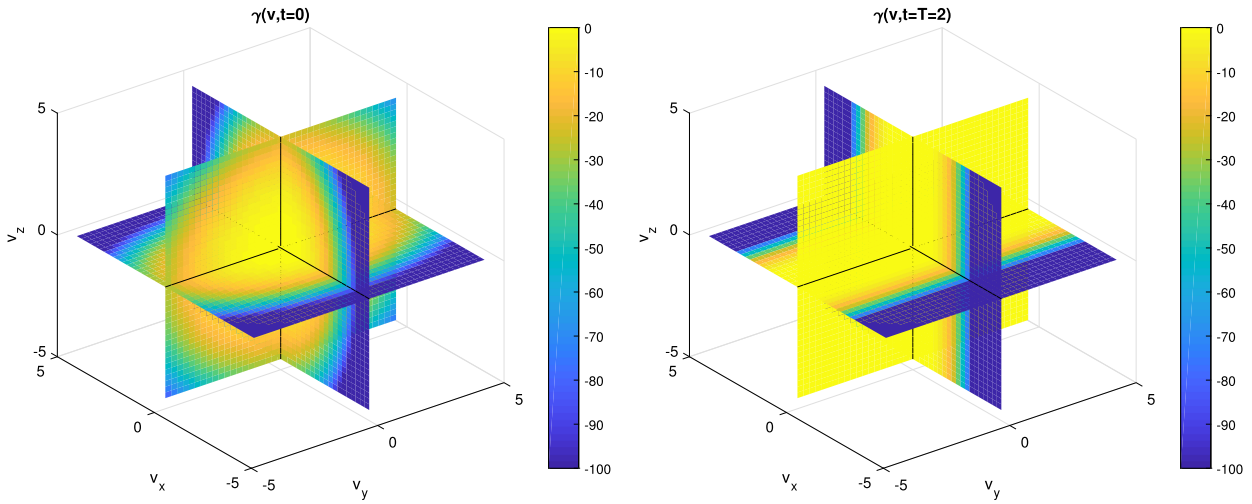

Fig. A.8. Slices of the function $\gamma(v,t)$ at $t=0$ (left) and $t=T=2$ (right) from the backward evolution of equation (A.1) using the numerical scheme (A.2). Here, $\gamma(v,t=T)=-r(v)=-v_x^4$, $n_{\text{grid}}=40$, $n_\varphi=n_\theta=10$ and $\Delta t=0.1$.

We rewrite the integral over σ as an integral over two angles, $\varphi\in[-\pi,\pi]$ and $\theta\in[0,\pi]$, and thus replace $d\sigma$ with $\sin\theta d\theta d\varphi = d(-\cos\theta)d\varphi$. We discretize φ with n_φ points, and then $\varphi_j=-\pi+2j\pi/n_\varphi$, $j=\{0,1,...,n_\varphi-1\}$. Similarly, we discretize $\cos\theta$ with n_θ points, and then $(\cos\theta)_h=-1+1/n_\theta+2h/n_\theta$, $h=\{0,1,...,n_\theta-1\}$. Thus,

$$\sigma_{j,h}=(\cos\varphi_j(\sin\theta)_h,\sin\varphi_j(\sin\theta)_h,(\cos\theta)_h),\ \text{where}\ (\sin\theta)_h=\sqrt{1-(\cos\theta)_h^2}\ \text{and}\ \Delta\sigma=4\pi/(n_\varphi n_\theta).$$

Finally, we discretize the integrals as sums over the grid $\{v_{i_x,i_y,i_z},\varphi_j,(\cos\theta)_h\}$, $i_x,i_y,i_z=\{0,1,...,n_{\text{grid}}-1\}$, $j=\{0,1,...,n_\varphi-1\}$, $h=\{0,1,...,n_\theta-1\}$ and use the backward Euler scheme for the time derivatives to obtain the following numerical scheme:

$$\begin{aligned}&\frac{\gamma(v_{i_x,i_y,i_z},t_k)-\gamma(v_{i_x,i_y,i_z},t_{k+1})}{\Delta t}\\&=2\sum_{i'_x,i'_y,i'_z=0}^{n_{\text{grid}}-1}\left(\sum_{j=0}^{n_\varphi-1}\sum_{h=0}^{n_\theta-1}\gamma(v',t_{k+1})\right)f(v_{i'_x,i'_y,i'_z},t_{k+1})\frac{\Delta\sigma(\Delta v)^3}{4\pi}\\&-\sum_{i'_x,i'_y,i'_z=0}^{n_{\text{grid}}-1}\gamma(v_{i'_x,i'_y,i'_z},t_{k+1})f(v_{i'_x,i'_y,i'_z},t_{k+1})(\Delta v)^3-\gamma(v_{i_x,i_y,i_z},t_{k+1}),\end{aligned}\tag{A.2}$$

where $v'=1/2(v_{i_x,i_y,i_z}+v_{i'_x,i'_y,i'_z})+1/2|v_{i_x,i_y,i_z}-v_{i'_x,i'_y,i'_z}|\sigma_{j,h}$ and we used our assumptions that $q(\sigma)=1/(4\pi)$ and $\rho=1$, and thus $\mu=\rho\int_{S^2}q(\sigma)d\sigma=1$. Since post-collision velocity v' does not always fall onto the grid points of the velocity space, we approximate the function value $\gamma(v',t_{k+1})$ by linear interpolation using the set of function values $\gamma(v_{i_x,i_y,i_z},t_{k+1})$. Fig. A.8 shows the function $\gamma(v,t)$ computed according to the numerical scheme (A.2) at $t=T=2$ and $t=0$ using $\gamma(v,t=T)=-r(v)=-v_x^4$ and $n_{\text{grid}}=40$, $n_\varphi=n_\theta=10$, $\Delta t=0.1$.

To compute the gradients of the objective function with respect to the parameter α, we use the values of $\gamma(v,0)$ and the derivative of the initial distribution with respect to α; see (20). For the initial distribution (49) where the parameter $\alpha=T_p^0$, $p\in\{x,y,z\}$, the derivative of the initial distribution is computed following (56). After approximating the integral in (20) by the Riemann sum over the grids, we have

$$\begin{aligned}\frac{\partial J_1}{\partial T_p^0}&=-\int\gamma(v,0)\frac{\partial f_0(v;T_p^0)}{\partial T_p^0}dv=-\int\gamma(v,0)\left(\frac{v_p^2}{T_p^0}-1\right)\frac{1}{2T_p^0}f_0(v)dv\\&\approx\sum_{i_x,i_y,i_z=0}^{n_{\text{grid}}-1}\gamma(v_{i_x,i_y,i_z},0)\left(\frac{v_{i_p}^2}{T_p^0}-1\right)\frac{1}{2T_p^0}f_0(v_{i_x,i_y,i_z})(\Delta v)^3.\end{aligned}$$

Appendix B. Numerical methods comparison. Memory requirements, error scaling and operation count

B.1. Memory requirements

(i) The forward DSMC simulations are done using the Nanbu–Babobsky scheme with N particles. Thus, we need $3N \times 8 = 24N$ bytes in double-precision arithmetic to store the velocity of the particles. At each time step, there are $N_c = N\mu\Delta t$ particles that collide, which is a small fraction of N. Thus, we disregard the temporary variables used for the computation of the collisions in calculating the memory requirements. We also override the particle velocities to not use extra memory for the new velocities at each time step. To compute the gradients by the finite difference, we run several simulations with slightly different values of the parameter α. Since these simulations can be done sequentially, it does not increase the overall memory requirements.

(ii) The adjoint DSMC method requires $3N \times 8 = 24N$ bytes for the storage of $\boldsymbol{\gamma}_{k,i}$ in addition to the memory requirements from the forward DSMC propagation. During the forward DSMC simulation, we also need to store the following information about the collisions that happen at each time step. For each colliding pair, we store

- the indices of the two colliding particles, 4 bytes each if stored in the UInt32 format allowing for values in $[0, 2^{32} - 1]$; since $2^{32} \approx 4.3 \times 10^9 > 10^8$, it is good enough for our purposes;
- the σ vector in $\mathbb{R}^3$, 8 bytes for each component of the vector;
- the unit collision direction $\frac{v-v_1}{|v-v_1|} \in \mathbb{R}^3$, 8 bytes for each component.

In total, the memory requirements are 56 bytes per colliding pair or 28 bytes per colliding particle. Overall, we need $28N_c$ 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 $28N_cM = 28N\mu\Delta tT/\Delta t = 28N\mu T$. It becomes comparable to the $24N$ bytes needed for the particle storage in the forward DSMC when $28N\mu T > 24N$, or when roughly $\mu T > 1$. The total amount of extra memory required for the backward propagation is $24N + 28N\mu T$ bytes. In our simulations, we use $\mu = 1$, $T = 2$, so the total amount of memory required for the adjoint DSMC was about four times the amount of memory required for the forward DSMC.

(iii) The DSMC-type scheme for the continuous adjoint equation (19a) requires $1N \times 8 = 8N$ extra bytes for the storage of $\gamma(v_{k,i}, t_k)$. Like the forward DSMC, we do not consider temporary storage needed for the colliding particles at each time step since they constitute only a small fraction of the overall number of particles and the new data for $\gamma(v_i, t_k)$ overrides the old one. We also do not store $v_{k,i}$ at every time step during the forward solve since we can always recover the velocity particles when marching back from $t = T$ to $t = 0$ based on equation (6) with the stored collision parameters σ. In this method, we also need to store all of the same information about colliding particles in the forward DSMC simulation as in the adjoint DSMC method, adding extra $28N\mu T$ bytes needed for storage total of $8N + 28N\mu T$ bytes.

(iv) For the direct discretization of the integrals in the adjoint equation (19a), we perform the forward simulation with the standard DSMC method. Afterwards, we need to convert velocity samples $\{v_{k,i}\}_{i=1}^N$ to a distribution function $f(v, t_k)$ via a histogram at every time step t_k during the forward propagation. With n_{grid} being the number of the grid points in each direction of the v-space, we have in total n_{grid}^3 grid points, which requires $8n_{\text{grid}}^3 M$ bytes of memory to store the distribution function $f(v, t)$ at every time step. In our simulations, $n_{\text{grid}} = 40$ considering the high computational complexity. With $n_{\text{grid}}^3 = 64000 \approx N/1000$ for $N = 10^8$, we use ≈ 1000 particles per Δv^3 box on average. During the backward solve, we need to store n_{grid}^3 values of the function $\gamma(v, t)$. For the right-hand side of (A.2), we can subtract one term of the right-hand side at a time to save memory. Therefore, in total, we need $8n_{\text{grid}}^3(M + 2)$ bytes for the backward solve. With $n_{\text{grid}}^3 = N/1000$ and $M = 20$, we get $8n_{\text{grid}}^3(M + 2) = 0.176N$ bytes which is much smaller than the number of bytes required in the forward DSMC simulation.

B.2. Error scaling

To compare the accuracy, we assume that only one simulation is done for each value of the parameter α. We use the forward Euler time integration for all the methods, so the time-integration error is $\mathcal{O}(\Delta t)$ for all the four approaches. As it is hard to find all the constants in the error estimates, we only provide error scaling here.

(i) Due to the Monte Carlo representation of the distribution function $f(v, t)$, we have a spatial error $\mathcal{O}(1/\sqrt{N})$ in computing the moments. For the gradient calculation, due to the finite difference scheme, we have errors $\mathcal{O}((\Delta\alpha)^2) + \mathcal{O}(1/(\Delta\alpha\sqrt{N}))$ contributed from (53) and (54).

(ii) Similar to the finite difference scheme, due to the Monte Carlo representation of the distribution function $f(v, t)$, we have a spatial error $\mathcal{O}(1/\sqrt{N})$.

(iii) Again, the Monte Carlo representations of $f(v, t)$ and $\gamma_i(t_k)$ contribute to spatial error $\mathcal{O}(1/\sqrt{N})$. The interpolation part of the DSMC-type scheme for the adjoint equation does not introduce significant extra error since the local error of a linear interpolation scales like $\propto (\overline{\Delta v})^2 \propto 1/N^{2/3} < 1/N^{1/2}$, where $\overline{\Delta v}$ is a characteristic distance between particles.

(iv) In (19a), functions under the integral sign are smooth, non-singular (including the constant kernel q) and thus can be considered periodic in the φ-space, $\cos\theta$-space and the v-space (up to numerical round-off errors on the boundaries of the domain), so we can achieve effectively exponential convergence from the integral itself. However, since we have to compute

$\gamma(v', t_{k+1})$ in (A.2) using interpolation, the overall order of convergence is limited by the order of interpolation. The linear interpolation introduces an error of size $\mathcal{O}((\Delta v)^2)$. The forward simulation for (13) can be done by a deterministic method by computing the integrals in a similar way here as to how we handle the continuous adjoint equation, which can then achieve a similar error of $\mathcal{O}((\Delta v)^2)$. However, in this paper, we stick with computing $f(v,t)$ by simply converting the empirical distribution represented by the N particles to a histogram at each time step. Thus, we get an additional error of $\mathcal{O}(1/\sqrt{N})$ by the forward DSMC.

B.3. Operation count

In this subsection, we provide operation counts to illustrate the performance of the methods. The same as before, we only provide the dominant scaling for the operation counts.

(i) Colliding two particles takes 47 operations per collision pair in our code or 23.5 operations per colliding particle. That is, $23.5N_cM = 23.5N\mu T$ total operations per simulation. To compute the gradient via the finite difference scheme, we need to run at least $D_\alpha + 1$ simulations with slightly different values of the parameter α, where D_α is the dimensionality of α. Overall, to compute the gradient $\frac{dJ_1}{d\alpha}$, we need $(D_\alpha + 1) \times 23.5N\mu T$ operations.

(ii) The adjoint DSMC algorithm takes 26 operations to collide two γ-particles together. There are $N_c/2$ pairs of particles collided at every time step, and M total backward steps. We get the total operation count $26/2N_cM = 13N\mu T$. Also, 6 more operations are needed to compute the post-collision unit direction σ per a collision pair, and 3 more operations are needed to compute the unit collision direction $(v - v_1)\hat{}$ during the forward DSMC propagation for each collision pair. They are used later in the backward adjoint DSMC step. In total, we need $17.5N\mu T$ operations per backward simulation in the adjoint DSMC algorithm.

(iii) In the DSMC-type scheme for the continuous adjoint equation (19a), we only need to perform 2 operations and 1 interpolation per particle at each time step. Interpolation is linear, but it is done here on a scattered (non-structured) set of data. We use a built-in MATLAB function to interpolate the scattered data, which uses a Delaunay triangulation on the scattered sample points to find neighboring points to the target location and perform the interpolation. The operation count of Delaunay triangulation scales like $\mathcal{O}(N\log(N))$. This is why this method is relatively slow compared to the adjoint DSMC scheme. For the interpolation itself, we believe it is reasonable to assume that we need about 20 operations per point. As a result, it brings us roughly up to $(20N_c + CN\log(N))M = (20 + C\log(N)/\Delta t)N\mu T$ operations per simulation. Besides, in the DSMC-type scheme, we need velocities $v_{k,i}$ at each time step t_k. If we do not save them during the forward propagation, we have to recover the velocity particles at a particular time from the final velocities $\{v_{F,i}\}_{i=1}^N$ by marching backward in time. This is done in the same way as how we backpropagate the adjoint particle $\boldsymbol{\gamma}_{k,i}$ from t_{k+1} to t_k; see (6). The back-propagation of the velocity particles requires an extra $17.5N\mu T$ operations per simulation. The total number of operations is roughly $(40 + C\log(N)/\Delta t)N\mu T$ for the DSMC-type scheme.

(iv) Computing an integral over dv and $d\sigma$ is the most expensive part at each time step for the direct discretization of (19a). Inside the integral, we need to compute $\gamma(v')$ using interpolation from the grid points in the v-space to the point v'. Again, we assume 20 operations are needed per grid point for the interpolation and the computation of v'. The total number of operations needed to compute $\gamma(v, t_k)$ from $\gamma(v, t_{k+1})$ in (A.2) per grid point in the v-space is about $(21n_\varphi n_\theta + 4)n_{\text{grid}}^3$. Since there are n_{grid}^3 grid points in the v-space grid and M time steps, the total operation count per simulation is about $(21n_\varphi n_\theta + 4)n_{\text{grid}}^6 M$. Assuming $n_{\text{grid}}^3 = N/1000$ and $n_\varphi = n_\theta = 10$, we get $(21n_\varphi n_\theta + 4)n_{\text{grid}}^6 M = 0.0021N^2M$. There are drastically more operations in the direct discretization than the adjoint DSMC algorithm as a result of the $\mathcal{O}(N^2)$ scaling where typically $N \geq 10^6$.

References

[1] F. Abraham, M. Behr, M. Heinkenschloss, The effect of stabilization in finite element methods for the optimal boundary control of the Oseen equations, Finite Elem. Anal. Des. 41 (2004) 229–251.

[2] G. Albi, M. Herty, L. Pareschi, Kinetic description of optimal control problems and applications to opinion consensus, Commun. Math. Sci. 13 (2015) 1407–1429.

[3] G. Albi, L. Pareschi, M. Zanella, Boltzmann-type control of opinion consensus through leaders, Philos. Trans. R. Soc. A, Math. Phys. Eng. Sci. 372 (2014) 20140138.

[4] R.J. Alonso, Boltzmann-Type Equations and Their Applications, IMPA, 2015.

[5] L. Arkeryd, Stability in l^1 for the spatially homogeneous Boltzmann equation, Arch. Ration. Mech. Anal. 103 (1988) 151–167.

[6] H. Babovsky, R. Illner, A convergence proof for Nanbu's simulation method for the full Boltzmann equation, SIAM J. Numer. Anal. 26 (1989) 45–65.

[7] J.T. Betts, S.L. Campbell, Discretize then optimize, in: Mathematics for Industry: Challenges and Frontiers, 2005, pp. 140–157.

[8] P.L. Bhatnagar, E.P. Gross, M. Krook, A model for collision processes in gases, I: small amplitude processes in charged and neutral one-component systems, Phys. Rev. 94 (1954) 511.

[9] L.T. Biegler, O. Ghattas, M. Heinkenschloss, B. van Bloemen Waanders, Large-scale PDE-constrained optimization: an introduction, in: Large-Scale PDE-Constrained Optimization, Springer, 2003, pp. 3–13.

[10] G. Bird, Direct simulation and the Boltzmann equation, Phys. Fluids 13 (1970) 2676–2681.

[11] A.V. Bobylev, E. Mossberg, On some properties of linear and linearized Boltzmann collision operators for hard spheres, Kinet. Relat. Models 1 (2008) 521.

[12] A.V. Bobylev, K. Nanbu, Theory of collision algorithms for gases and plasmas based on the Boltzmann equation and the Landau–Fokker–Planck equation, Phys. Rev. E 61 (2000) 4576–4586, https://doi.org/10.1103/PhysRevE.61.4576.

[13] J. Burkardt, M. Gunzburger, J. Peterson, Insensitive functionals, inconsistent gradients, spurious minima, and regularized functionals in flow optimization problems, Int. J. Comput. Fluid Dyn. 16 (2002) 171–185.
[14] R.E. Caflisch, The Boltzmann equation with a soft potential, Commun. Math. Phys. 74 (1980) 71–95.
[15] R.E. Caflisch, Monte Carlo and quasi-Monte Carlo methods, Acta Numer. 7 (1998) 1–49.
[16] Y. Cao, S. Li, L. Petzold, R. Serban, Adjoint sensitivity analysis for differential-algebraic equations: the adjoint DAE system and its numerical solution, SIAM J. Sci. Comput. 24 (2003) 1076–1089.
[17] M. Caponigro, M. Fornasier, B. Piccoli, E. Trélat, Sparse stabilization and optimal control of the Cucker–Smale model, Math. Control Relat. Fields 3 (2013) 447–466.
[18] C. Cercignani, Mathematical Methods in Kinetic Theory, Springer, 1969.
[19] C. Cercignani, The Boltzmann equation and its applications, Appl. Math. Sci. (1988).
[20] G. Chavent, M. Dupuy, P. Lemmonier, History matching by use of optimal theory, Soc. Pet. Eng. J. 15 (1975) 74–86.
[21] Y. Cheng, I.M. Gamba, K. Ren, Recovering doping profiles in semiconductor devices with the Boltzmann–Poisson model, J. Comput. Phys. 230 (2011) 3391–3412.
[22] M. Choulli, P. Stefanov, Inverse scattering and inverse boundary value problems for the linear Boltzmann equation, Commun. Partial Differ. Equ. 21 (1996) 763–785.
[23] A.A. Dragulescu, Applications of physics to economics and finance: money, income, wealth, and the stock market, preprint, arXiv:cond-mat/0307341, 2003.
[24] M. Fornasier, B. Piccoli, F. Rossi, Mean-field sparse optimal control, Philos. Trans. R. Soc. A, Math. Phys. Eng. Sci. 372 (2014) 20130400.
[25] M. Fornasier, F. Solombrino, Mean-field optimal control, ESAIM Control Optim. Calc. Var. 20 (2014) 1123–1152.
[26] K. Ghobadi, N.S. Nedialkov, T. Terlaky, On the discretize then optimize approach, Preprint for Industrial and Systems Engineering, 2009.
[27] P. Glasserman, Monte Carlo Methods in Financial Engineering, vol. 53, Springer Science & Business Media, 2013.
[28] W.W. Hager, Runge-Kutta methods in optimal control and the transformed adjoint system, Numer. Math. 87 (2000) 247–282.
[29] M. Hinze, A. Rösch, Discretization of optimal control problems, in: Constrained Optimization and Optimal Control for Partial Differential Equations, Springer, 2012, pp. 391–430.
[30] L. Holway, Kinetic theory of shock structure using an ellipsoidal distribution function, Rarefied Gas Dyn. 1 (1966) 193–215.
[31] R.Y. Lai, G. Uhlmann, Y. Yang, Reconstruction of the collision kernel in the nonlinear Boltzmann equation, preprint, arXiv:2003.09549, 2020.
[32] Y. LeCun, D. Touresky, G. Hinton, T. Sejnowski, A theoretical framework for back-propagation, in: Proceedings of the 1988 Connectionist Models Summer School, CMU, Morgan Kaufmann, Pittsburgh, PA, 1988, pp. 21–28.
[33] J. Liu, Z. Wang, Non-commutative discretize-then-optimize algorithms for elliptic PDE-constrained optimal control problems, J. Comput. Appl. Math. 362 (2019) 596–613.
[34] F.A. Longstaff, E.S. Schwartz, Valuing American options by simulation: a simple least-squares approach, Rev. Financ. Stud. 14 (2001) 113–147.
[35] J.C. Maxwell, IV. On the Dynamical Theory of Gases, Philos. Trans. R. Soc. Lond. 157 (1867) 49–88.
[36] P.R. McGillivray, D. Oldenburg, Methods for calculating Fréchet derivatives and sensitivities for the non-linear inverse problem: a comparative study 1, Geophys. Prospect. 38 (1990) 499–524.
[37] S. Mohamed, M. Rosca, M. Figurnov, A. Mnih, Monte Carlo gradient estimation in machine learning, preprint, arXiv:1906.10652, 2019.
[38] K. Nanbu, Direct simulation scheme derived from the Boltzmann equation, I: monocomponent gases, J. Phys. Soc. Jpn. 49 (1980) 2042–2049.
[39] J. Nocedal, S. Wright, Numerical Optimization, Springer Science & Business Media, 2006.
[40] L. Pareschi, G. Russo, An introduction to Monte Carlo method for the Boltzmann equation, in: ESAIM: Proceedings, EDP Sciences, 2001, pp. 35–75.
[41] R.E. Plessix, A review of the adjoint-state method for computing the gradient of a functional with geophysical applications, Geophys. J. Int. 167 (2006) 495–503.
[42] A. Sato, T. Yamada, K. Izui, S. Nishiwaki, S. Takata, A topology optimization method in rarefied gas flow problems using the Boltzmann equation, J. Comput. Phys. 395 (2019) 60–84.
[43] E. Shakhov, Generalization of the Krook kinetic relaxation equation, Fluid Dyn. 3 (1968) 95–96.
[44] G. Toscani, Kinetic models of opinion formation, Commun. Math. Sci. 4 (2006) 481–496.
[45] J. Tsitsiklis, B. Van Roy, Optimal stopping of Markov processes: Hilbert space theory, approximation algorithms, and an application to pricing high-dimensional financial derivatives, IEEE Trans. Autom. Control 44 (1991) 1840–1851.
[46] C. Villani, A review of mathematical topics in collisional kinetic theory, in: Handbook of Mathematical Fluid Dynamics, vol. 1, 2002, pp. 3–8.
[47] C. Wang, T. Lin, R. Caflisch, B.I. Cohen, A.M. Dimits, Particle simulation of Coulomb collisions: comparing the methods of Takizuka & Abe and Nanbu, J. Comput. Phys. 227 (2008) 4308–4329.