Journal of Computational Physics 375 (2018) 519–539

Contents lists available at ScienceDirect

Journal of Computational Physics

www.elsevier.com/locate/jcp

Parallel tensor methods for high-dimensional linear PDEs

Arnout M.P. Boelens [a], Daniele Venturi [b], Daniel M. Tartakovsky [a,*]

[a] Department of Energy Resources Engineering, Stanford University, Stanford, CA 94305, United States of America
[b] Department of Applied Mathematics and Statistics, UC Santa Cruz, Santa Cruz, CA 95064, United States of America

ARTICLE INFO

Article history:
Received 21 March 2018
Received in revised form 21 August 2018
Accepted 31 August 2018
Available online xxxx

Keywords:
PDF equation
Method of distributions
High-dimensional PDEs

ABSTRACT

High-dimensional partial-differential equations (PDEs) arise in a number of fields of science and engineering, where they are used to describe the evolution of joint probability functions. Their examples include the Boltzmann and Fokker–Planck equations. We develop new parallel algorithms to solve such high-dimensional PDEs. The algorithms are based on canonical and hierarchical numerical tensor methods combined with alternating least squares and hierarchical singular value decomposition. Both implicit and explicit integration schemes are presented and discussed. We demonstrate the accuracy and efficiency of the proposed new algorithms in computing the numerical solution to both an advection equation in six variables plus time and a linearized version of the Boltzmann equation.

1. Introduction

High-dimensional partial-differential equations (PDEs) arise in a number of fields of science and engineering. For example, they play an important role in modeling rarefied gas dynamics [1], stochastic dynamical systems [2,3], structural dynamics [4], turbulence [5–7], biological networks [8], and quantum systems [9,10]. In the context of kinetic theory and stochastic dynamics, high-dimensional PDEs typically describe the evolution of a (joint) probability density function (PDF) of system states, providing macroscopic, continuum descriptions of Langevin-type stochastic systems. Classical examples of such PDEs include the Boltzmann equation [1] and the Fokker–Planck equation [11]. More recently, PDF equations have been used to quantify uncertainty in model predictions [12].

High dimensionality and resulting computational complexity of PDF equations can be mitigated by using particle-based methods [13,14]. Well known examples are direct simulation Monte Carlo (DSMC) [15] and the Nambu–Babovsky method [16]. Such methods preserve physical properties of a system and exhibit high computational efficiency (they scale linearly with the number of particles), in particular in simulations far from statistical equilibrium [17,18]. Moreover, these methods have relatively low memory requirements since the particles tend to concentrate where the distribution function is not small. However, the accuracy of particle methods may be poor and their predictions are subject to significant statistical fluctuations [18–20]. Such fluctuations need to be post-processed appropriately, for example by using variance reduction techniques. Also, relevance and applicability of these numerical strategies to PDEs other than kinetic equations are not clear.

To overcome these difficulties, several general-purpose algorithms have been recently proposed to compute the numerical solution to rather general high-dimensional linear PDEs. The most efficient techniques, however, are problem specific [14,21–23]. For example, in the context of kinetic methods, there is an extensive literature concerning the (six-

* Corresponding author.
E-mail address: tartakovsky@stanford.edu (D.M. Tartakovsky).

https://doi.org/10.1016/j.jcp.2018.08.057
0021-9991/

dimensional) Boltzmann equation and its solutions close to statistical equilibrium. Deterministic methods for solving such problems include semi-Lagrangian schemes and discrete velocity models. The former employ a fixed computational grid, account for transport features of the Boltzmann equation in a fully Lagrangian framework, and usually adapt operator splitting. The latter use a regular grid in velocity and a discrete collision operator on the points of the grid that preserves the main physical properties (see sections 3 and 4 in [14] for an in-depth review of semi-Lagrangian methods and discrete velocity models for kinetic transport equations, respectively).

We develop new parallel algorithms to solve high-dimensional partial differential equations based on numerical tensor methods [24]. The algorithms we propose are based on canonical and hierarchical tensor expansions combined with alternating least squares and hierarchical singular value decomposition. The key element that opens the possibility to solve high-dimensional PDEs numerically with tensor methods is that tensor approximations proved to be capable of representing function-related N-dimensional data arrays of size Q^N with log-volume complexity $\mathcal{O}(N\log(Q))$. Combined with traditional deterministic numerical schemes (e.g., spectral collocation method [25], these novel representations allow one to compute the solution to high-dimensional PDEs using low-parametric rank-structured tensor formats.

This paper is organized as follows. In Section 2 we discuss tensor representations of N-dimensional functions. In particular, we discuss canonical tensor decomposition and hierarchical tensor networks, including hierarchical Tucker and tensor train expansions. We also address the problem of computing tensor expansion via alternating least squares. In Section 3 we develop several algorithms that rely on numerical tensor methods to solve high-dimensional PDEs. Specifically, we discuss schemes based on implicit and explicit time integration, combined with dimensional splitting, alternating least squares and hierarchical singular value decomposition. In Section 4 we demonstrate the effectiveness and computational efficiency of the proposed new algorithms in solving both an advection equation in six variables plus time and a linearized version of the Boltzmann equation.

2. Tensor decomposition of high-dimensional functions

Consider a multivariate scalar field $f(\boldsymbol{z}) : D \subseteq \mathbb{R}^N \to \mathbb{R}$. In this section we briefly review effective representations of f based on tensor methods. In particular, we discuss the canonical tensor decomposition and hierarchical tensor methods, e.g., tensor train and hierarchical Tucker expansions.

2.1. Canonical tensor decomposition

The canonical tensor decomposition of the multivariate function $f(\boldsymbol{z}) : D \subseteq \mathbb{R}^N \to \mathbb{R}$ is a series expansion of the form

$$f(\boldsymbol{z}) \simeq \sum_{l=1}^{r} \prod_{i=1}^{N} G_i^l(z_i), \tag{1}$$

where r is called a separation rank, and the one-dimensional functions $G_i^l(z_i) : \mathbb{R} \to \mathbb{R}$ are usually represented relative to a known basis $\{\phi_1, ..., \phi_Q\}$,

$$G_i^l(z_i) = \sum_{k=1}^{Q} \beta_{ik}^l \phi_k(z_i), \qquad i = 1, \ldots, N. \tag{2}$$

Although general/computable theorems relating a prescribed accuracy of the representation of $f(\boldsymbol{z})$ to the value of the separation rank r are still lacking, there are cases where the expansion (1) is exponentially more efficient than one would expect a priori [26]. In general, the number of modes Q in (2) is dimension-dependent, i.e., one can expand the solution in different variables z_i with different number of modes Q_i. To simplify the notation, we take Q to be identical in all variables, such that $Q_i = Q$ for all $i = 1, \ldots, N$.

Alternating least squares (ALS) formulation. Development of robust and efficient algorithms to compute (1) to any desired accuracy is still a relatively open question (see [27–30,21] for recent progresses). Computing the tensor components $G_k^l(z_k)$ usually relies on (greedy) optimization techniques, such as alternating least squares (ALS) [31,32,27,26] or regularized Newton methods [28], which are only locally convergent [33], i.e., the final result may depend on the initial condition of the algorithm.

In the least-squares setting, the tensor components $G_j^l(z_j)$ are computed by minimizing some norm of the residual,

$$R(\boldsymbol{z}) = f(\boldsymbol{z}) - \sum_{l=1}^{r} \prod_{i=1}^{N} G_i^l(z_i), \tag{3}$$

with respect to all rNQ degrees of freedom $\boldsymbol{\beta} = [\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_N]$, with

$$\boldsymbol{\beta}_1 = (\beta_{11}^1, \ldots, \beta_{1Q}^1, \ldots, \beta_{11}^r, \ldots, \beta_{1Q}^r)^\top, \quad \cdots, \quad \boldsymbol{\beta}_N = (\beta_{N1}^1, \ldots, \beta_{NQ}^1, \ldots \beta_{N1}^r, \ldots \beta_{NQ}^r)^\top. \tag{4}$$

The vector $\boldsymbol{\beta}_i$ $(i = 1, \ldots, N)$ collects the z_i-dependent degrees of freedom of all functions $\{G_i^1(z_i), \ldots, G_i^r(z_i)\}$. If $f(\boldsymbol{z})$ is periodic in the hypercube $D = [-b, b]^N$ $(b > 0)$, and choosing the L^2 norm, this minimization gives a variational principle

$$\min_{\boldsymbol{\beta}} \|R(\boldsymbol{z})\|_{L^2}^2 , \tag{5}$$

where

$$\|R(\boldsymbol{z})\|_{L_2}^2 \equiv \int_{-b}^{b} \cdots \int_{-b}^{b} R(\boldsymbol{z})^2 \mathrm{d}\boldsymbol{z} = \int_{-b}^{b} \cdots \int_{-b}^{b} \left[f(\boldsymbol{z}) - \sum_{n=1}^{r} \prod_{k=1}^{N} \left(\sum_{s=1}^{Q} \beta_{ks}^n \phi_s(z_k) \right) \right]^2 \mathrm{d}\boldsymbol{z}. \tag{6}$$

In the alternating least-squares (ALS) paradigm, the non-convex optimization problem (5) is split into a sequence of convex low-dimensional convex optimization problems

$$\min_{\boldsymbol{\beta}_1} \|R\|_{L_2}^2 , \quad \cdots , \quad \min_{\boldsymbol{\beta}_N} \|R\|_{L_2}^2 . \tag{7}$$

This sequence is not equivalent to the full problem (5) because, in general, it does not allow one to compute the global minimizer of (5) [33–36]. The Euler–Lagrange equations associated with (7) are of the form[1]

$$\boldsymbol{A}_j \boldsymbol{\beta}_j = \boldsymbol{g}_j \qquad j = 1, \ldots, N, \tag{9}$$

where

$$\boldsymbol{A}_j = \begin{bmatrix} A_{j11}^{11} \cdots A_{j1Q}^{11} & A_{j11}^{12} \cdots A_{j1Q}^{12} & \cdots & A_{j11}^{1r} \cdots A_{j1Q}^{1r} \\ \vdots & \vdots & \vdots & \vdots \\ A_{jQ1}^{11} \cdots A_{jQQ}^{11} & A_{jQ1}^{12} \cdots A_{jQQ}^{12} & \cdots & A_{jQ1}^{1r} \cdots A_{jQQ}^{1r} \\ \vdots & \vdots & \vdots & \vdots \\ A_{j11}^{r1} \cdots A_{j1Q}^{r1} & A_{j11}^{r2} \cdots A_{j1Q}^{r2} & \cdots & A_{j11}^{rr} \cdots A_{j1Q}^{rr} \\ \vdots & \vdots & \vdots & \vdots \\ A_{jQ1}^{r1} \cdots A_{jQQ}^{r1} & A_{jQ1}^{r2} \cdots A_{jQQ}^{r2} & \cdots & A_{jQ1}^{rr} \cdots A_{jQQ}^{rr} \end{bmatrix}, \qquad \boldsymbol{g}_j = \begin{bmatrix} g_{j1}^1 \\ \vdots \\ g_{jQ}^1 \\ \vdots \\ g_{j1}^r \\ \vdots \\ g_{jQ}^r \end{bmatrix}, \tag{10}$$

and

$$A_{jhs}^{ln} = \int_{-b}^{b} \phi_h(z_j)\phi_s(z_j)\mathrm{d}z_j \prod_{\substack{k=1 \\ k\neq j}}^{N} \int_{-b}^{b} G_k^l(z_k) G_k^n(z_k)\mathrm{d}z_k, \tag{11}$$

$$g_{jh}^n = \int_{-b}^{b} \cdots \int_{-b}^{b} f(\boldsymbol{x})\phi_h(x_j) \prod_{\substack{k=1 \\ k\neq j}}^{N} G_k^n(x_k)\mathrm{d}z_1 \cdots \mathrm{d}z_N. \tag{12}$$

The matrices $\boldsymbol{A}_j$ are symmetric, positive definite and of size $rQ \times rQ$.

Convergence of the ALS algorithm. The ALS algorithm described above is an alternating optimization scheme, i.e., a nonlinear block Gauss–Seidel method ([37], §7.4). There is a well–developed local convergence theory for this type of methods [37, 35]. In particular, it can be shown that ALS is locally equivalent to the linear block Gauss–Seidel iteration applied to the Hessian matrix. This implies that ALS is linearly convergent in the iteration number [33], provided that the Hessian of the residual is positive definite (except on a trivial null space associated with the scaling non-uniqueness of the canonical tensor decomposition). The last assumption may not be always satisfied. Therefore, convergence of the ALS algorithm cannot be granted in general. Another potential issue of the ALS algorithm is the poor conditioning of the matrices $\boldsymbol{A}_j$ in (9), which can addressed by regularization [31,32]. The canonical tensor decomposition (1) in N dimensions has relatively small memory requirements. In fact, the number of degrees of freedom that we need to store is rNQ, where r is the separation rank, and Q is the number of degrees of freedom employed in each tensor component $G_k^l(z_k)$. Despite the relatively low-memory requirements, it is often desirable to employ scalable parallel versions the ALS algorithm [29,38] to compute the canonical tensor expansion (1) (see Appendix B)

[1] Recall that minimizing the residual (3) with respect to β_{hj}^n is equivalent to impose orthogonality relative to the space spanned by the functions

$$\phi_h(z_j) \prod_{\substack{k=1 \\ k\neq j}}^{N} G_k^n(z_k). \tag{8}$$

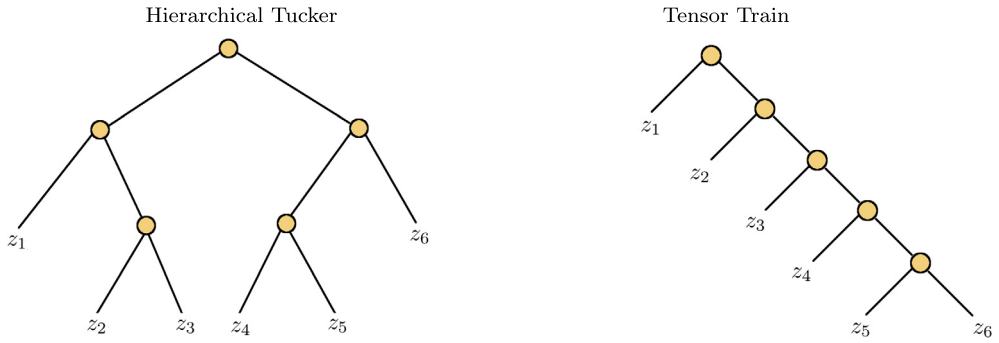

Fig. 1. Graph representation of the hierarchical Tucker (HT) and tensor train (TT) decomposition of a six-dimensional function $f(z_1, \ldots, z_6)$.

2.2. Hierarchical tensor methods

Hierarchical tensor methods [39,40] were introduced to mitigate the dimensionality problem in the core tensor of the classical Tucker decomposition [41]. A key idea is to perform a sequence of Schmidt decompositions (or multivariate SVDs [42,43]) until the approximation problem is reduced to a product of one-dimensional functions/vectors. To illustrate the method in a simple way, consider a six-dimensional function $f(\boldsymbol{z}) = f(z_1, \ldots, z_6)$. We first split the variables as (z_1, z_2, z_3) and (z_4, z_5, z_6) through one Schmidt decomposition [44] as

$$f(\boldsymbol{z}) = \sum_{i_7, i_8 = 1}^{r} A^{\{1\}}_{i_7 i_8} T^{\{1,2,3\}}_{i_7}(z_1, z_2, z_3) T^{\{4,5,6\}}_{i_8}(z_4, z_5, z_6). \tag{13}$$

Then we decompose $T^{\{1,2,3\}}_{i_7}(z_1, z_2, z_3)$ and $T^{\{4,5,6\}}_{i_8}(z_4, z_5, z_6)$ further by additional Schmidt expansions to obtain

$$\begin{aligned} f(\boldsymbol{z}) &= \sum_{i_7, i_8 = 1}^{r} A^{\{1\}}_{i_7 i_8} \sum_{i_1, i_9 = 1}^{r} A^{\{2\}}_{i_7 i_1 i_9} T^{\{1\}}_{i_1}(z_1) T^{\{2,3\}}_{i_9}(z_2, z_3) \sum_{i_4, i_{10} = 1}^{r} A^{\{3\}}_{i_8 i_4 i_{10}} T^{\{4\}}_{i_4}(z_4) T^{\{5,6\}}_{i_{10}}(z_5, z_6), \\ &= \sum_{i_1, \cdots, i_6 = 1}^{r} C_{i_1 \cdots i_6} T^{\{1\}}_{i_1}(z_1) T^{\{2\}}_{i_2}(z_2) T^{\{3\}}_{i_3}(z_3) T^{\{4\}}_{i_4}(z_4) T^{\{5\}}_{i_5}(z_5) T^{\{6\}}_{i_6}(z_6). \end{aligned} \tag{14}$$

The 6-dimensional *core tensor*[2] is explicitly obtained as

$$C_{i_1 \cdots i_6} = \sum_{i_7, i_8 = 1}^{r} A^{\{1\}}_{i_7 i_8} \sum_{i_9 = 1}^{r} A^{\{2\}}_{i_7 i_1 i_9} A^{\{4\}}_{i_9 i_2 i_3} \sum_{i_{10} = 1}^{r} A^{\{3\}}_{i_8 i_4 i_{10}} A^{\{5\}}_{i_{10} i_5 i_6}. \tag{16}$$

The procedure just described forms the foundation of hierarchical tensor methods. The key element is that the core tensor $C_{i_1 \cdots i_6}$ resulting from this procedure is always factored as a *product of at most three-dimensional matrices*. This is also true in arbitrary dimensions. The tensor components $T^{\{k\}}_{i_j}$ and the factors of the core tensor (16) can be effectively computed by employing hierarchical singular value decomposition [42,49,43]. Alternatively, one can use an optimization framework that leverages recursive subspace factorizations [50]. If we single out one variable at the time and perform a sequential Schmidt decomposition of the remaining variables we obtain the so-called *tensor-train* (TT) decomposition [51,36]. Both tensor train and hierarchical tensor expansions can be conveniently visualized by *graphs* (see Fig. 1). This is done by adopting the following standard rules: i) a node in a graph represents a tensor in as many variables as the number of the edges connected to it, ii) connecting two tensors by an edge represents a tensor contraction over the index associated with a certain variable.

[2] A diagonalization of the core tensor in (14) would minimize the number of terms in the series expansion. Unfortunately, this is impossible for a tensor with dimension larger than 2 (see, e.g., [45,46,41,47]) and, for complex tensors, [48]). A closer look at the canonical tensor decomposition (1) reveals that such an expansion is in the form of a fully diagonal high-order Schmidt decomposition, i.e.,

$$f(z_1, \ldots, z_N) = \sum_{l=1}^{r} C_{l \cdots l} G^l_1(z_1) \cdots G^l_N(z_N). \tag{15}$$

The fact that diagonalization of $C_{l_1 \cdots l_6}$ is impossible in dimension larger than 2 implies that it is impossible to compute the canonical tensor decomposition of f by standard linear algebra techniques.

Efficient algorithms to perform basic operations between hierarchical tensors, such as addition, orthogonalization, rank reduction, scalar products, multiplication, and linear transformations, are discussed in [52,41,49,53]. Parallel implementations of such algorithms were recently proposed in [54,55]. Methods for reducing the computational cost of tensor trains are discussed in [56,57,53]. Applications to the Vlasov kinetic equation can be found in [58–60].

3. Numerical approximation of high-dimensional PDEs

In this section we develop efficient numerical methods to integrate high-dimensional linear PDEs of the form

$$\frac{\partial f(\boldsymbol{z},t)}{\partial t} = L(\boldsymbol{z}) f(\boldsymbol{z},t), \tag{17}$$

where $L(\boldsymbol{z})$ is a linear operator. The algorithms we propose are based on numerical tensor methods and appropriate rank-reduction techniques, such as hierarchical singular value decomposition [42,55] and alternating least squares [29,28]. To begin with, we assume that $L(\boldsymbol{z})$ is a separable linear operator with separation rank r_L, i.e., an operator of the form

$$L(\boldsymbol{z}) = \sum_{l=1}^{r_L} \alpha_l L_1^l(z_1) \cdots L_N^l(z_N). \tag{18}$$

For each i and j, $L_i^j(z_i)$ is a linear operator acting only on variable z_i. As an example, the operator

$$L(z_1,z_2) = z_1^2 \frac{\partial}{\partial z_1} - \sin(z_2) \frac{\partial^2}{\partial z_1 \partial z_2}, \tag{19}$$

is separable with separation rank $r_L = 2$ in dimension $N = 2$. It can be written in the form (18) if we set $\alpha_1 = \alpha_2 = 1$ and

$$L_1^1(z_1) = z_1^2 \frac{\partial}{\partial z_1}, \quad L_2^1(z_2) = 1, \quad L_1^2(z_1) = \frac{\partial}{\partial z_1}, \quad L_2^2(z_2) = -\sin(z_2) \frac{\partial}{\partial z_2}. \tag{20}$$

3.1. Tensor methods with implicit time stepping

Let us discretize the PDE in (17) in time by using the Crank–Nicolson method. To this end, consider an evenly-spaced grid $t_n = n\Delta t$ $(n = 0, 1, \ldots)$ with time step Δt. Defining $f_n(\boldsymbol{z}) \equiv f(\boldsymbol{z}, t_n)$, we write the discretization scheme as (e.g., [21])

$$A(\boldsymbol{z}) f_{n+1}(\boldsymbol{z}) = B(\boldsymbol{z}) f_n(\boldsymbol{z}) + \Delta t \tau_{n+1}(\boldsymbol{z}), \tag{21}$$

where

$$A = I - \frac{\Delta t}{2} L(\boldsymbol{z}) \quad \text{and} \quad B = I + \frac{\Delta t}{2} L(\boldsymbol{z}). \tag{22}$$

Here I denotes the identity operator, and $\tau_{n+1}(\boldsymbol{z})$ represents the local truncation error of the Crank–Nicolson method at time t_{n+1} [61]. It follows from the definition of $L(\boldsymbol{z})$ in (18) that both $A(\boldsymbol{z})$ and $B(\boldsymbol{z})$ are separable operators of the form

$$A = \sum_{q=0}^{r_L} \eta_q E_1^q(z_1) \cdots E_N^q(z_N), \qquad B = \sum_{q=0}^{r_L} \zeta_q E_1^q(z_1) \cdots E_N^q(z_N), \tag{23}$$

where $\eta_0 = \zeta_0 = 1$, $E_j^0(z_j) = 1$ $(j = 1, \ldots, N)$,

$$\eta_q = -\frac{\Delta t}{2}, \qquad \zeta_q = -\eta_q, \qquad E_j^q(z_j) = L_j^q(z_j), \qquad q = 1, \ldots, r_L, \quad j = 1, \ldots, N. \tag{24}$$

A substitution of the canonical tensor decomposition[3]

$$\hat{f}_{n+1}(\boldsymbol{z}) = \sum_{l=1}^{r} \prod_{k=1}^{N} G_k^l(z_k, t_{n+1}) \tag{25}$$

[3] Recall that the functions $G_k^l(z_k, t_n)$ are in the form

$$G_k^l(z_k, t_n) = \sum_{s=1}^{Q} \beta_{ks}^l(t_n) \phi_s(z_k),$$

where $\beta_{ks}^l(t_n)$ $(l = 1, \ldots, r,\ k = 1, \ldots, N,\ s = 1, \ldots, Q)$ are the degrees of freedom.

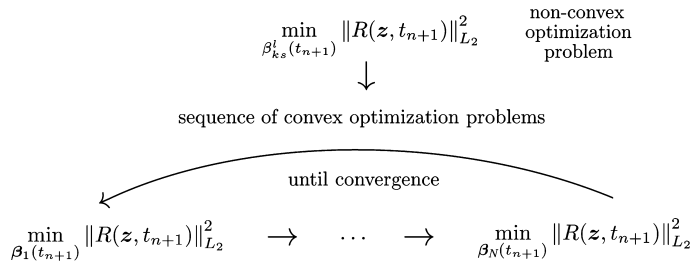

Fig. 2. Sketch of the alternating least squares (ALS) algorithm for the minimization of the norm of the residual $R(\boldsymbol{z}, t_{n+1})$ defined in equation (26).

into (21) yields the residual

$$R(\boldsymbol{z}, t_{n+1}) = A(\boldsymbol{z})\hat{f}_{n+1}(\boldsymbol{z}) - B(\boldsymbol{z})\hat{f}_n(\boldsymbol{z}), \tag{26}$$

in which the local truncation error $\Delta t \tau_{n+1}$ is embedded into $R(\boldsymbol{z}, t_{n+1})$.

Next, we minimize the residual using the alternating least squares algorithm, described in Section 3.1, to obtain the solution at time t_{n+1}. To obtain an efficient minimizer of (26), we split the optimization problem

$$\min_{\boldsymbol{\beta}(t_{n+1})} \|R(\boldsymbol{z}, t_{n+1})\|^2_{L_2} \tag{27}$$

into a sequence of optimization problems of smaller dimension, which are solved iteratively. For each dimension, the residual is minimized independently of the other dimensions; this allows for parallelization of the problem [29]. Once convergence is reached for each individual dimension, the results are combined and the minimization procedure is repeated until global convergence is reached for all dimensions (Appendix B). To this end, we define

$$\boldsymbol{\beta}_k(t_n) = [\beta^1_{k1}(t_n), \ldots, \beta^1_{kQ}(t_n), \ldots, \beta^r_{k1}(t_n), \ldots, \beta^r_{kQ}(t_n)]^\top \qquad k = 1, \ldots, N. \tag{28}$$

The vector $\boldsymbol{\beta}_k(t_n)$ collects the degrees of freedom representing the solution functional along z_k at time t_n, i.e., the set of functions $\{G^1_k(z_k, t_n), ..., G^r_k(z_k, t_n)\}$. Minimization of (27) with respect to independent variations of $\boldsymbol{\beta}_k(t_{n+1})$ yields a sequence of *convex* optimization problems (see Fig. 2)

$$\min_{\boldsymbol{\beta}_1(t_{n+1})} \|R(\boldsymbol{z}, t_{n+1})\|^2_{L_2}, \quad \min_{\boldsymbol{\beta}_2(t_{n+1})} \|R(\boldsymbol{z}, t_{n+1})\|^2_{L_2}, \quad \cdots, \quad \min_{\boldsymbol{\beta}_N(t_{n+1})} \|R(\boldsymbol{z}, t_{n+1})\|^2_{L_2}. \tag{29}$$

This set of equations defines the alternating least-squares (ALS) method.

The Euler–Lagrange equations, which identify stationary points of (29), are linear systems in the form

$$\boldsymbol{M}^L_q \boldsymbol{\beta}_q(t_{n+1}) = \boldsymbol{M}^R_q \boldsymbol{\beta}_q(t_n) \qquad q = 1, \ldots, N, \tag{30}$$

where

$$\boldsymbol{M}^L_q = \sum_{e,z=0}^{r_L} K^L_{ez} \left[\bigcirc_{\substack{k=1 \\ k\neq q}}^{N} \hat{\boldsymbol{\beta}}_k(t_{n+1})^T \boldsymbol{E}^{ez}_k \hat{\boldsymbol{\beta}}_k(t_{n+1}) \right]^T \otimes \left[\boldsymbol{E}^{ez}_q\right]^T, \tag{31}$$

$$\boldsymbol{M}^R_q = \sum_{e,z=0}^{r_L} K^R_{ez} \left[\bigcirc_{\substack{k=1 \\ k\neq q}}^{N} \hat{\boldsymbol{\beta}}_k(t_n)^T \boldsymbol{E}^{ez}_k \hat{\boldsymbol{\beta}}_k(t_{n+1}) \right]^T \otimes \left[\boldsymbol{E}^{ez}_q\right]^T, \tag{32}$$

and

$$\left[\boldsymbol{E}^{ez}_q\right]_{sh} = \int_{-b}^{b} E^e_q(a)\phi_s(a)E^z_q(a)\phi_h(a)\mathrm{d}a. \tag{33}$$

In (31) and (32), $\bigcirc$ denotes the Hadamard matrix product; $\otimes$ is the Kronecker matrix product; $\hat{\boldsymbol{\beta}}_k$ is the matricization of $\boldsymbol{\beta}_k$, i.e.,

$$\hat{\boldsymbol{\beta}}_k(t_n) = \begin{bmatrix} \beta^1_{k1}(t_n) & \cdots & \beta^r_{k1}(t_n) \\ \vdots & \ddots & \vdots \\ \beta^1_{kQ}(t_n) & \cdots & \beta^r_{kQ}(t_n) \end{bmatrix}; \tag{34}$$

$\boldsymbol{E}_q^{ez}$ are $Q \times Q$ matrices; and K_{ez}^L and K_{ez}^R are entries of the matrices

$$\boldsymbol{K}^L = \boldsymbol{\eta}\boldsymbol{\eta}^T \quad \text{and} \quad \boldsymbol{K}^R = \boldsymbol{\eta}\boldsymbol{\zeta}^T, \tag{35}$$

where $\boldsymbol{\eta}$ and $\boldsymbol{\zeta}$ are column vectors with the entries η_q and ζ_q defined in (23). The ALS algorithm, combined with canonical tensor representations, effectively reduces evaluation of N-dimensional integrals to evaluation of the sum of products of one-dimensional integrals.

Summary of the algorithm. Many of the integrals in (33) differ only by a prefactor, in which case the number of unique integrals is reduced to a small number (4 in the case of the Boltzmann–BGK equation discussed in Section 4.2). Thus, it is convenient to precompute a map between such integrals and any entry in (33). Such a map allows us to rapidly compute each matrix $\boldsymbol{E}_q^{ez}$ in (31) and (32), and therefore to efficiently build the matrix system. Next, we compute the canonical tensor decomposition of the initial condition $f(\boldsymbol{z}, 0)$ by applying the methods described in Section 2.1. This yields the set of vectors $\{\boldsymbol{\beta}_1(t_0), ..., \boldsymbol{\beta}_N(t_0)\}$, from which the matrices $\boldsymbol{M}_1^L$ and $\boldsymbol{M}_1^R$ in (31) and (32) are built. To this end, we need an initial guess for $\{\boldsymbol{\beta}_1(t_1), ..., \boldsymbol{\beta}_N(t_1)\}$ given, e.g., by $\{\boldsymbol{\beta}_1(t_0), ..., \boldsymbol{\beta}_N(t_0)\}$ or its small random perturbation. With $\boldsymbol{M}_1^L$, $\boldsymbol{M}_1^R$ in place, we solve the linear system in (30) and update $\boldsymbol{\beta}_1(t_1)$. Finally, we recompute $\boldsymbol{M}_2^L$ and $\boldsymbol{M}_2^R$ (with the updated $\boldsymbol{\beta}_1(t_1)$) and solve for $\boldsymbol{\beta}_2(t_1)$. This process is repeated for $q = 3, \ldots, N$ and iterated among all the variables until convergence (see Fig. 2). The ALS iterations are said to converge if

$$\left\| \boldsymbol{\beta}_k^{(j)}(t_n) - \boldsymbol{\beta}_k^{(j-1)}(t_n) \right\|_2^2 \leq \epsilon_{\text{Tol}}, \qquad j = 1, \ldots, N_{\max}, \tag{36}$$

where j here denotes the iteration number whose maximum value is $N_{\max}$, and ϵ_{Tol} is a prescribed tolerance on the increment of such iterations. Parallel versions of the ALS algorithm associated with canonical tensor decompositions were recently proposed in [29,38] (see also Appendix B). We emphasize that the use of a hierarchical tensor series instead of (25) (see Section (2.2)) also allows one to compute the unknown degrees of the expansion by using parallel ALS algorithms. This question was recently addressed in [54].

3.2. Tensor methods with explicit time stepping

Let us discretize PDE (17) in time by using any explicit time stepping scheme, for example the second-order Adams–Bashforth scheme [61]

$$f_{n+2}(\boldsymbol{z}) = f_{n+1}(\boldsymbol{z}) + \frac{\Delta t}{2} L(\boldsymbol{z}) \left(3 f_{n+1}(\boldsymbol{z}) - f_n(\boldsymbol{z})\right) + \Delta t \tau_{n+2}(\boldsymbol{z}). \tag{37}$$

In this setting, the only operations needed to compute f_{n+2} with tensor methods are: i) addition, ii) application of a (separable) linear operator L, and iii) rank reduction,[4] the last operation being the most important among the three [41, 49,31]. Rank reduction can be achieved, e.g., by hierarchical singular value decomposition [42,55] or by alternating least squares [29].

From a computational perspective, it is useful to split the sequence of tensor operations yielding f_{n+2} in (37) into simple tensor operations followed by rank reduction. For example, $L(3f_{n+1} - f_n)$ might be evaluated as follows: i) compute $w_{n+1} = f_{n+1} - f_n$; ii) perform rank reduction on w_{n+1}; iii) compute Lw_{n+1} and perform rank reduction again. This allows one to minimize the memory requirements and overall computational cost.[5] Unfortunately, splitting tensor operations into sequences of tensor operations followed by rank reduction can produce severe cancellation errors. In some cases this problem can be mitigated. For example, an efficient and robust algorithm, which allows us to split sums and rank reduction operations, was recently proposed in [49] in the context of hierarchical Tucker formats. The algorithm leverages the block diagonal structure arising from addition of hierarchical Tucker tensors.

4. Numerical results

In this section, we demonstrate the accuracy and effectiveness of the numerical tensor methods discussed above. Two case studies are considered: an advection equation in six dimensions plus time and the Boltzmann–BGK equation. In both cases, the computational domain is a tensor product of one-dimensional intervals, i.e., a hypercube with periodic boundary conditions. This allows us to use Fourier series as basis functions.

[4] On the other hand, the use of implicit time-discretization schemes requires one to solve linear systems with tensor methods [55,49,31].

[5] Recall that adding two tensors with rank r_1 and r_2 usually yields a tensor of rank $r_1 + r_2$.

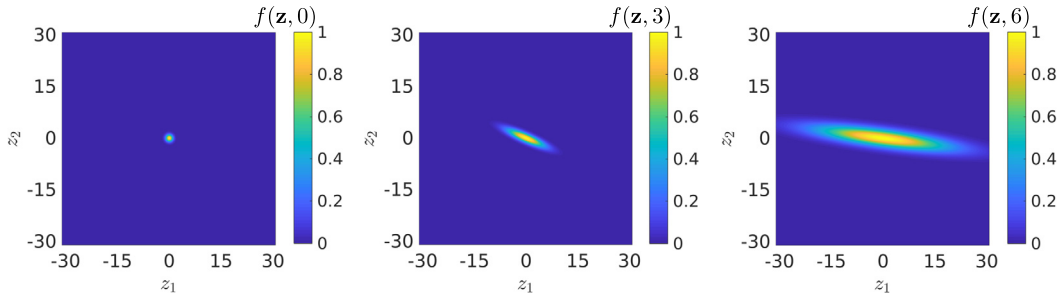

Fig. 3. Temporal snapshots (at $t = 0$, 3, and 6) of the multivariate PDF $f(\mathbf{z}, t)$, given by the analytical solution (39) and (40), in $N = 2$. The stable spiral at the origin of the characteristic system attracts every curve in the phase space and ultimately yields $f = 1$ everywhere after a transient.

4.1. Advection equation

Consider an initial-value problem

$$\frac{\partial f}{\partial t} + \sum_{j=1}^{N} \left(\sum_{k=1}^{N} C_{jk} z_k \right) \frac{\partial f}{\partial z_j} = 0, \qquad f(z_1, \ldots, z_N, 0) = f_0(z_1, \ldots, z_N), \tag{38}$$

where N is the number of independent variables forming the coordinate vector $\boldsymbol{z} = (z_1, \ldots, z_N)^\top$, and C_{jk} are components of a given matrix of coefficients $\boldsymbol{C}$. The analytical solution to this initial-value problem is computed with the method of characteristics as [62]

$$f(\boldsymbol{z}, t) = f_0\left(\mathrm{e}^{-t\boldsymbol{C}} \boldsymbol{z}\right). \tag{39}$$

We choose the initial condition f_0 to be a fully separable product of exponentials,

$$f_0(\boldsymbol{z}) = \mathrm{e}^{-\|\boldsymbol{z}\|_2^2}. \tag{40}$$

The time dynamics of the solution (39)–(40) is entirely determined by the matrix $\boldsymbol{C}$. In particular, if $\boldsymbol{C}$ has eigenvalues with positive real part then the matrix exponential $\exp(-t\boldsymbol{C})$ is a contraction map. In this case, $f(\boldsymbol{z}, t) \to 1$ as $t \to \infty$, at each point $\boldsymbol{z} \in \mathbb{R}^N$. Fig. 3 exhibits the analytical solution (39) generated by a 2×2 matrix $\boldsymbol{C}$, whose eigenvalues are complex conjugate with positive real part. The side length of the hypercube that encloses any level set of the solution at time t depends on the number of dimensions N. In particular, the side-length of the hypercube that encloses the set $\{\boldsymbol{z} \in \mathbb{R}^N \mid f(\boldsymbol{z}, t) \geq \epsilon\}$ is given by

$$\frac{1}{2}\sqrt{\frac{-\log(\epsilon)}{N\lambda_{\min}}} \leq b \leq \frac{1}{2}\sqrt{\frac{-\log(\epsilon)}{\lambda_{\min}}}, \tag{41}$$

where $\lambda_{\min}$ is the smallest eigenvalue of the matrix $\exp(t\boldsymbol{C})^\top \exp(t\boldsymbol{C})$. Equation (38) can be written in the form (17), with separable operator L (separation rank N^2). Specifically, the one-dimensional operators $L_j^l(z_j)$ appearing in (18) are explicitly defined in Table 1.

We used tensor methods, both canonical tensor decomposition and hierarchical tensor methods, and explicit time stepping (Section 3.2) to solve numerically (38) in the periodic hypercube[6] $[-60, 60]^N$. Each tensor component was discretized by using a Fourier spectral expansion with 600 nodes in each variable (e.g., basis functions in (2) or (14)). The accuracy of the numerical solution was quantified in terms of the time-dependent relative error

$$\epsilon_m(t) = \left| \frac{f(\boldsymbol{z}^*, t) - \hat{f}(\boldsymbol{z}^*, t)}{f(\boldsymbol{z}^*, t)} \right|, \tag{42}$$

where f is the analytical solution (39), $\hat{f}$ is the numerical solution obtained by using the canonical or hierarchical tensor methods with separation rank r.

Fig. 4 shows the relative pointwise error (42), computed at $\boldsymbol{z}^* = (h, h, \ldots, h)$ with $h = 0.698835274542439$, for different separation ranks r and different number of independent variables N. As expected, the accuracy of the numerical solution increases with the separation rank r. Also, the relative error increases with the number of dimensions N. It is worthwhile emphasizing that the rank reduction in canonical tensor decomposition at each time step is based on a randomized

[6] Such domain is chosen large enough to accommodate periodic (zero) boundary conditions in the integration period of interest.

Table 1
Advection equation (38). Ordering of the linear operators defined in (18).

q	α_q	L_1^q	L_2^q	$\cdots$	L_N^q
1	$-C_{11}$	$z_1\partial_{z_1}$	1	$\cdots$	1
2	$-C_{12}$	∂_{z_1}	z_2	$\cdots$	1
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
N	$-C_{1N}$	∂_{z_1}	1	$\cdots$	z_N
$N+1$	$-C_{21}$	z_1	∂_{z_2}	$\cdots$	1
$N+2$	$-C_{22}$	1	$z_2\partial_{z_2}$	$\cdots$	1
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$2N$	$-C_{2N}$	1	∂_{z_2}	$\cdots$	z_N
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
N^2-N+1	$-C_{N1}$	z_1	1	$\cdots$	∂_{z_N}
N^2-N+2	$-C_{N2}$	1	z_2	$\cdots$	∂_{z_N}
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
N^2	$-C_{NN}$	1	1	$\cdots$	$z_N\partial_{z_N}$

algorithm that requires initialization at each time-step. This means that results of simulations with the same nominal parameters may be different. On the other hand, tensor methods based on a hierarchical singular value decomposition, such as the hierarchical Tucker decomposition [42,55], do not suffer from this issue. The separation rank of both canonical and hierarchical tensor methods is computed adaptively up to the maximum value $r_{\max}$ specified in the legend of Fig. 4. In two dimensions, the CP-ALS algorithm yields the similar error plots for $r_{\max}=8$ and $r_{\max}=12$. This is because $r<8$ in both cases, and throughout the simulation up to $t=1$. The difference between these results is due to the random initialization required by the ALS algorithm at each time step.

4.2. Boltzmann–BGK equation

In this section we develop an accurate ALS-based algorithm to solve a linearized BGK equation (see Appendix A for details of its derivation),

$$\frac{\partial f}{\partial t} + \boldsymbol{v}\cdot\nabla_{\boldsymbol{x}} f = \nu I, \qquad I = f_{\mathrm{eq}}(\boldsymbol{v}) - f. \tag{43}$$

Here, $f(\boldsymbol{z},t)$ is a probability density function is six phase variables plus time; the coordinate vector $\boldsymbol{z}=(\boldsymbol{x},\boldsymbol{v})^\top$ is composed of three spatial dimensions $\boldsymbol{x}=(x_1,x_2,x_3)^\top$ and three components of the velocity vector $\boldsymbol{v}=(v_1,v_2,v_3)^\top$; f_{eq} denotes a locally Maxwellian distribution,

$$f_{\mathrm{eq}}(\boldsymbol{v}) = \frac{\rho}{(2\pi RT)^{3/2}}\exp\left(-\frac{\|\boldsymbol{v}\|_2^2}{2RT}\right), \tag{44}$$

with uniform gas density ρ, temperature T, and velocity $\boldsymbol{v}$; R is the gas constant; and $\nu=\kappa\rho T^{1-\mu}$ with positive constants κ and μ. The solution to the linearized Boltzmann–BGK equation (45) is also computable analytically, which provides us with a benchmark solution to check the accuracy of our algorithms.

As before we start by rewriting this equation in an operator form,

$$\frac{\partial f}{\partial t} = Lf + C \tag{45}$$

where

$$L(\boldsymbol{x},\boldsymbol{v}) = -\boldsymbol{v}\cdot\nabla_{\boldsymbol{x}} - \nu I, \qquad C(\boldsymbol{v}) = \nu f_{\mathrm{eq}}(\boldsymbol{v}). \tag{46}$$

Note that L is a *separable* linear operator with separation rank $r_L=4$, which can be rewritten in the form

$$L(\boldsymbol{z}) = \sum_{q=1}^{4}\alpha_q L_1^q(z_1)\cdots L_6^q(z_6), \tag{47}$$

for suitable one-dimensional linear operators $L_j^q(a_j)$ defined in Table 2. Similarly, $C(\boldsymbol{v})$ in (46) is separated as

$$C(\boldsymbol{v}) = C_1(v_1)C_2(v_2)C_3(v_3), \quad \text{where} \quad C_i(v_i) = \frac{(\nu\rho)^{1/3}}{(2\pi RT)^{1/2}}\exp\left(-\frac{v_i^2}{2RT}\right). \tag{48}$$

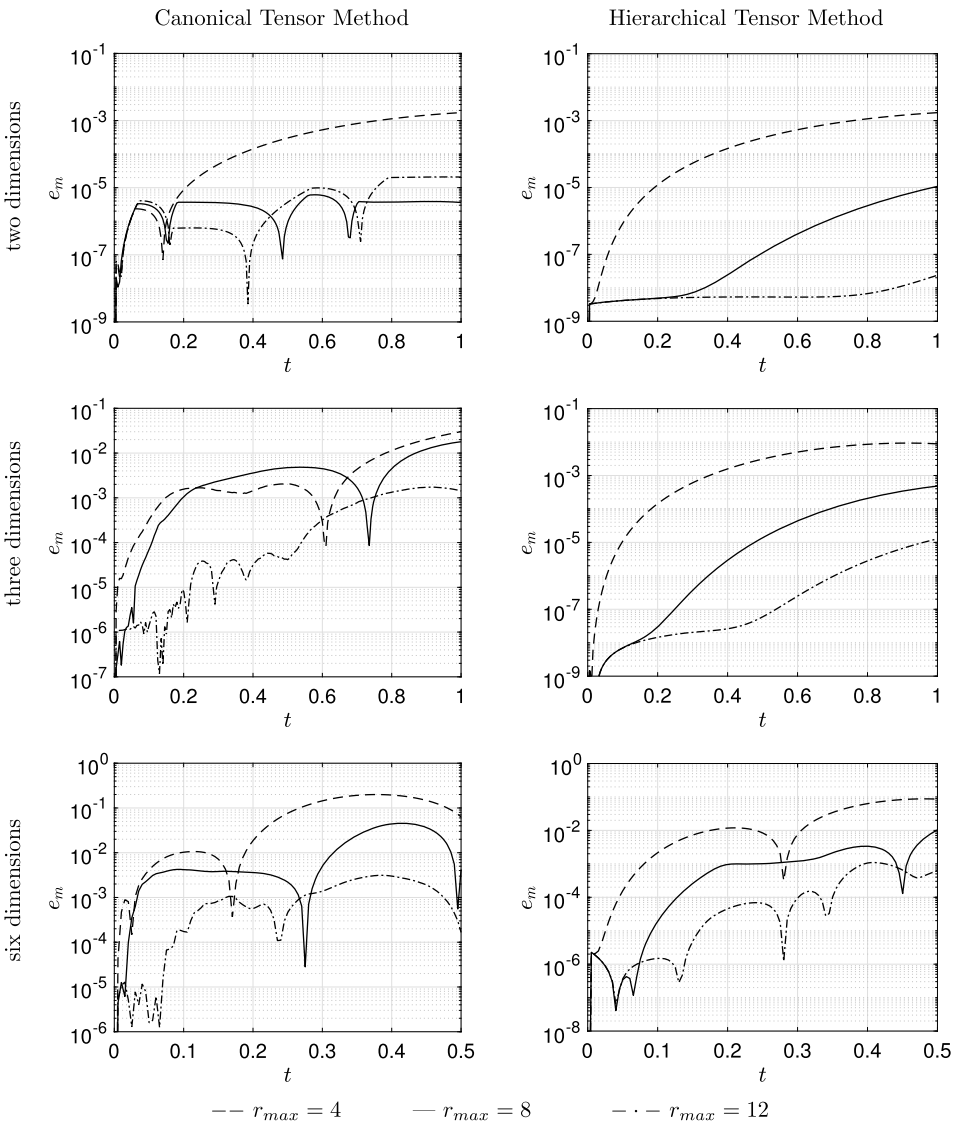

Fig. 4. Advection equation (38). Relative pointwise error (42) for several values of the separation ranks r and the number of independent variables N. These results are obtained by using the canonical and hierarchical tensor methods with explicit time stepping (see Section 3.2). The separation rank is computed adaptively at each time step up to the maximum value $r_{\max}$.

Table 2
Ordering of the linear operators defined in equation (47).

q	α_q	L_1^q	L_2^q	L_3^q	L_4^q	L_5^q	L_6^q
1	$-\nu$	1	1	1	1	1	1
2	-1	$\partial/\partial x_1$	1	1	v_1	1	1
3	-1	1	$\partial/\partial x_2$	1	1	v_2	1
4	-1	1	1	$\partial/\partial x_3$	1	1	v_3

From a numerical viewpoint, the solution to (45) can be represented by using any of the tensor series expansion we discussed in Section 2. In particular, hereafter we develop an algorithm based on canonical tensor decomposition, alternating least squares, and implicit time stepping (see Section 3.1). To this end, we discretize (45) in time with the Crank–Nicolson method to obtain

$$\left[I - \frac{\Delta t}{2} L\right] f_{n+1} = \left[I + \frac{\Delta t}{2} L\right] f_n + \Delta t C + \Delta t \tau_n, \tag{49}$$

Table 3
Ordering of the linear operators A and B defined in (23).

q	η_q	ζ_q	E_1^q	E_2^q	E_3^q	E_4^q	E_5^q	E_6^q
1	$\nu(1+\Delta t/2)$	$\nu(1-\Delta t/2)$	1	1	1	1	1	1
1	$\Delta t/2$	$-\Delta t/2$	$\partial/\partial x_1$	1	1	v_1	1	1
1	$\Delta t/2$	$-\Delta t/2$	1	$\partial/\partial x_2$	1	1	v_2	1
1	$\Delta t/2$	$-\Delta t/2$	1	1	$\partial/\partial x_3$	1	1	v_3

where $f_n(\boldsymbol{z}) = f(\boldsymbol{z}, t_n)$ and τ_n is the local truncation error of the Crank–Nicolson method at $t = t_n$ ([61], p. 499). This equation can be written in a compact notation as

$$Af_{n+1} = Bf_n + \Delta tC + \Delta t\tau_n, \tag{50}$$

where A and B are separable operators in the form (23), where all quantities are defined in Table 3.

A substitution of the canonical tensor decomposition[7]

$$\hat{f}_{n+1} = \sum_{l=1}^{r}\prod_{k=1}^{6} G_k^l(z_k, t_{n+1}) \tag{51}$$

into equation (21) yields the residual

$$R(\boldsymbol{z}, t_{n+1}) = A(\boldsymbol{z})\hat{f}_{n+1}(\boldsymbol{z}) - B(\boldsymbol{z})\hat{f}_n(\boldsymbol{z}) + \Delta tC(z_4, z_5, z_6), \tag{52}$$

which can be minimized by using the alternating least squares method, as we described in Section 3.1 to obtain the solution at time t_{n+1}.

Nonlinear Boltzmann–BGK model. The numerical tensor methods we discussed in Section 3.1 and Section 3.2 can be extended to compute the numerical solution of the fully nonlinear Boltzmann–BGK model

$$\frac{\partial f}{\partial t} + \boldsymbol{v}\cdot\nabla_{\boldsymbol{x}} f = \nu(\boldsymbol{x}, t)\left(f_{\text{eq}}(\boldsymbol{x}, \boldsymbol{v}, t) - f(\boldsymbol{x}, \boldsymbol{v}, t)\right). \tag{53}$$

Here, $f_{\text{eq}}(\boldsymbol{x}, \boldsymbol{v}, t)$ is the equilibrium distribution (A.8), while the collision frequency $\nu(\boldsymbol{x}, t)$ can be expressed, e.g., as in (A.14). From a mathematical viewpoint both quantities f_{eq} and ν are nonlinear functionals of the PDF $f(\boldsymbol{x}, \boldsymbol{v}, t)$ (see equations (A.9)–(A.11)). Therefore (53) is an advection equation driven by a nonlinear functional of $f(\boldsymbol{x}, \boldsymbol{v}, t)$. The evaluation of the integrals appearing in (A.9)–(A.11) poses no great challenge if we represent f as a canonical tensor (1) or as a hierarchical Tucker tensor (14). Indeed, such integrals can be factored as sums of product of one-dimensional integrals in a tensor representation. Moreover, computing the inverse of $\rho(\boldsymbol{x}, t)$ in (A.10) and (A.11) is relatively simple in a collocation setting. In fact, we just need to evaluate the full tensor $\rho(\boldsymbol{x}, t)$, on a three-dimensional tensor product grid and compute $1/\rho(\boldsymbol{x}, t)$ pointwise. If needed, we could then compute the tensor decomposition of $1/\rho(x, t)$, e.g., by using the canonical tensor representations and the ALS algorithm we discussed in Section 2.1. This allows us to evaluate the BGK collision operator and represent it in a tensor series expansion. The solution to the Boltzmann BGK equation can be then computed by operator splitting methods [63,64].

Initial and boundary conditions. To validate the proposed alternating least squares algorithm we set the initial condition to be either coincident with the homogeneous Maxwell–Boltzmann distribution (44), i.e.,

$$f(\boldsymbol{x}, \boldsymbol{v}, 0) = \frac{\rho}{(2\pi RT)^{3/2}}\exp\left(-\frac{\|\boldsymbol{v}\|_2^2}{2RT}\right) \tag{54}$$

or a slight perturbation of it. The PDF (54) is obviously separable with separation rank one. Moreover, we set the computational domain to be a periodic hypercube $[-b_x, b_x]^3 \times [-b_v, b_v]^3$. It is essential that such domain is chosen large enough to accommodate the support of the PDF at all times, thus preventing physically unrealistic correlations between the distribution function. More generally, it is possible to enforce other types of boundary conditions by using appropriate trial functions, or by a mixed approach [65–67] in the case of Maxwell boundaries.

[7] Recall that the functions $G_k^l(z_k, t_n)$ are in the form

$$G_k^l(z_k, t_n) = \sum_{s=1}^{Q}\beta_{ks}^l(t_n)\phi_s(z_k),$$

where $\beta_{ks}^l(t_n)$ ($l = 1, \ldots, r$, $k = 1, \ldots, 6$, $s = 1, \ldots, Q$) are the degrees of freedom.

Table 4
Simulation parameters we employed in the numerical approximation of the Boltzmann–BGK model. These parameters correspond to a kinetic simulation of dynamics of Argon in the periodic hypercube $[-b_x, b_x]^3 \times [-b_v, b_v]^3$. The time step in the implicit Crank–Nicolson method is Δt, while the tolerance in the ALS iterations at each time step is ϵ_{Tol}.

Parameter	Symbol	value
Temperature	T	300 K
Number density	n	$2.4143 \cdot 10^{25}\,\text{m}^{-3}$
Specific gas constant	R	$208\,\text{J kg}^{-1}\,\text{K}^{-1}$
Relaxation time	τ_R	0.40034 s
Collision frequency	ν	$1/\tau_R$
Position domain size	b_x	500 m
Velocity domain size	b_v	$5\sqrt{R_S T}$
Time step	Δt	$0.01\tau_R$
Number of iterations	N_{Iter}	1000
ALS tolerance	ϵ_{Tol}	10^{-8}
Series truncation	Q	11

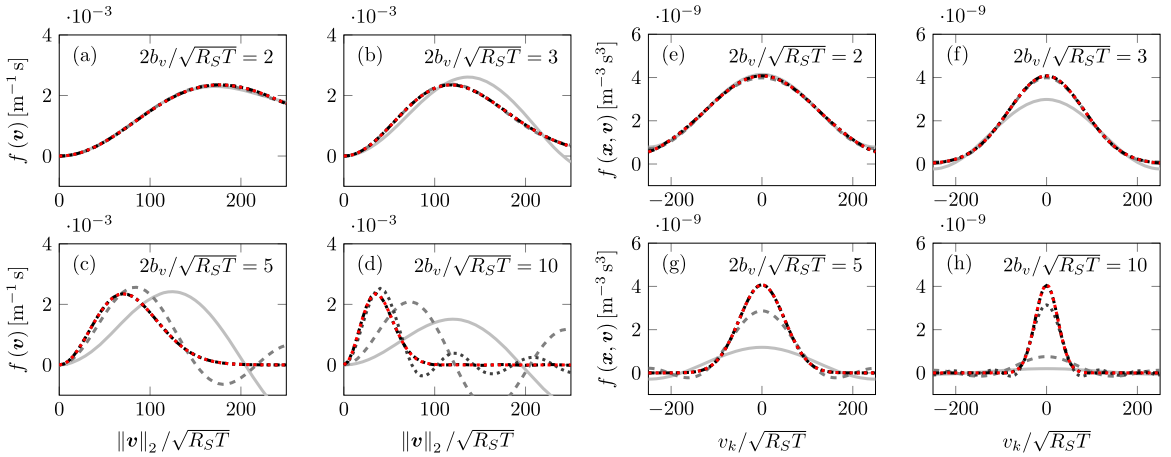

Fig. 5. One-particle Maxwell–Boltzmann distribution $f_{\text{eq}}(\boldsymbol{v})$ (▪▪▪▪▪) as a function of the dimensionless molecular speed $\|\boldsymbol{v}\|_2/\sqrt{RT}$ for different dimensionless domain sizes $b_v/\sqrt{RT}$. In each case, we demonstrate convergence of the canonical tensor decomposition (1) of f_{eq} as we increase the number of basis functions Q. Specifically, we plot the cases: $Q = 3$ (——), $Q = 5$ (– – –), $Q = 11$ (▪▪▪▪▪) and $Q = 19$ (▬▪▬▪).

Unless otherwise stated, all simulation parameters are set as in Table 4. Such setting corresponds to the problem of computing the dynamics of Argon within the periodic hypercube $[-b_x, b_x]^3 \times [-b_v, b_v]^3$. We are interested in testing our algorithms in two different regimes: i) steady state, and ii) relaxation to statistical equilibrium.

4.2.1. Canonical tensor decomposition of the Maxwell–Boltzmann distribution

We begin with a preliminary convergence study of the canonical tensor decomposition (1) applied to the one-particle Maxwell distribution $f_{\text{eq}}(\boldsymbol{v})$. Such study allows us to identify the size of the hypercube that includes the support of the Maxwellian equilibrium distribution. This is done in Fig. 5, where we plot the tensor expansion of $f_{\text{eq}}(\boldsymbol{v})$ as function of the non-dimensional molecular speed $\|\boldsymbol{v}\|_2/\sqrt{R_S T}$, for different number of basis functions Q in (2), and for various values of the dimensionless velocity domain size $b_v/\sqrt{RT}$. It is seen that for small values of $b_v/\sqrt{RT}$, a small number of basis functions Q is sufficient to capture the exact equilibrium distribution. At the same time however, small values of $b_v/\sqrt{RT}$ result in distribution functions with truncated tails, as can be seen in Figs. 5a, 5b, and 5e. Figs. 5c and 5g demonstrate that the dimensionless velocity domain size $b_v/\sqrt{RT} = 5$ is sufficient to avoid truncation of the tail of the distribution, and that a series expansion with $Q = 11$ in each variable is sufficient to accurately capture the equilibrium distribution. This justifies the choice of parameters in Table (4).

A more detailed analysis of the effects of the truncation order Q and the dimensionless velocity domain size $b_v/\sqrt{RT}$ on the approximation error in f_{eq} is done in Fig. 6a, where we plot the Normalized Mean Absolute Error (NMAE) versus $b_v/\sqrt{RT}$ and Q. The NMAE between two vectors $\boldsymbol{X}$ and $\boldsymbol{Y}$ of size N is defined as:

$$\text{NMAE} = \frac{1}{N}\frac{\|\boldsymbol{X} - \boldsymbol{Y}\|_1}{\max(\boldsymbol{X}) - \min(\boldsymbol{Y})}, \tag{55}$$

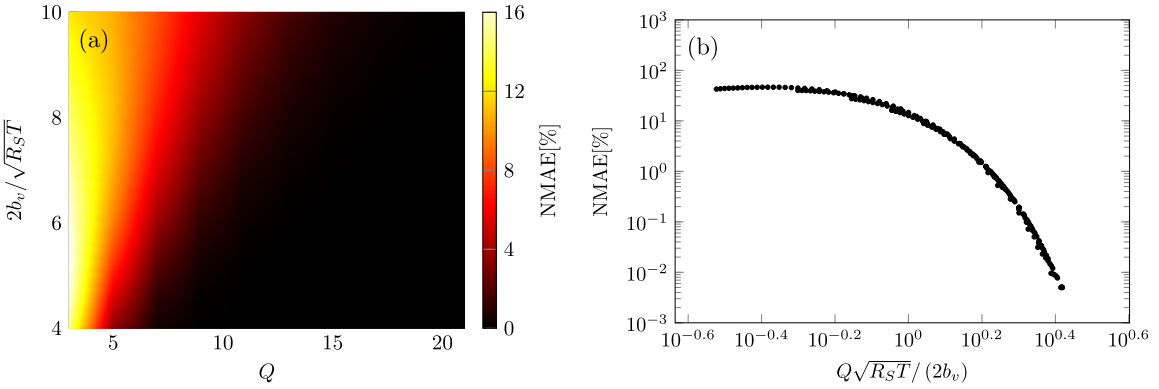

Fig. 6. (a) Normalized Mean Absolute Error (NMAE) (55) between the analytical and the canonical tensor series expansion of the Maxwell–Boltzmann equilibrium distribution versus Q and the dimensionless velocity domain size $b_v/\sqrt{RT}$. In Figure (b) we plot the NMAE in a log–log scale as function of $Q\sqrt{RT}/b_v$. We emphasize that the range of Q and $b_v/\sqrt{RT}$ is the same in both Figures.

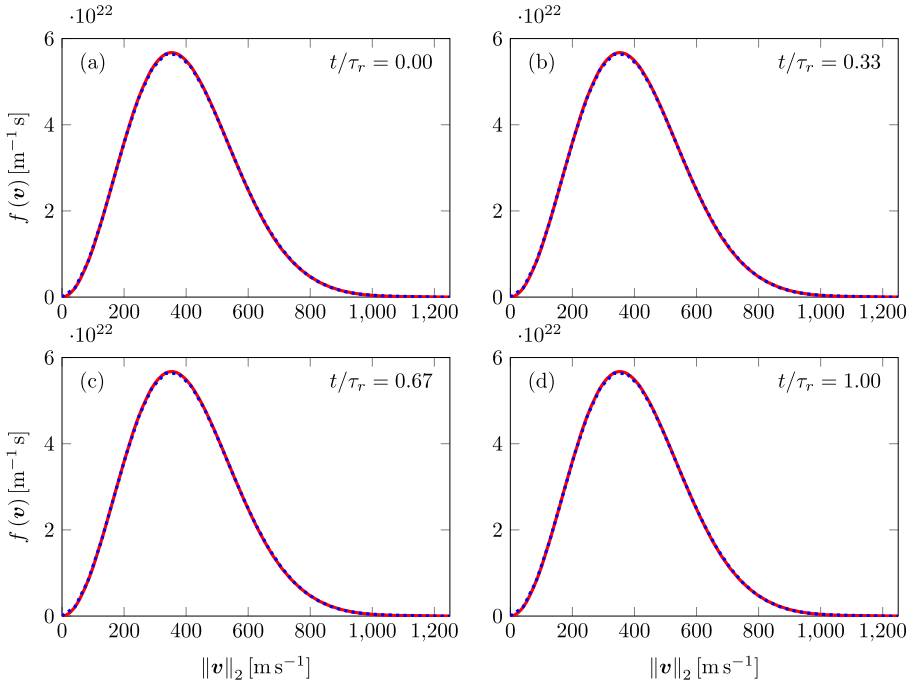

Fig. 7. Solution to the Boltzmann–BGK equation (43) at different dimensionless times t/τ_R (τ_R is the relaxation time in Table 4). Specifically, we plot the Maxwell–Boltzmann equilibrium distribution (——), its canonical tensor decomposition with $Q = 11$ modes, and its time evolution with the ALS algorithm we described in Section 3.1 (·····). It is seen that ALS has a stable fixed point at $f_{\mathrm{eq}}(\boldsymbol{v})$. This is an important test as, in general, ALS iterations are not granted to converge (e.g., [33]).

where $\|\cdot\|_1$ is the standard vector 1-norm. Fig. 6b exhibits the NMAE as a function of $Q\sqrt{RT}/b_v$. This results in a collapse of the data which suggests that if one wants to double the non-dimensional velocity domain size while maintaining the same accuracy, the series truncation order Q needs to be doubled as well. In addition, it can be seen that starting at $Q\sqrt{RT}/b_v \approx 1$ the error decays rapidly with increasing $Q\sqrt{RT}/b_v$.

4.2.2. Steady state simulations

We first test the Boltzmann–BGK solver we developed in Section 3.1 on an initial value problem where the initial condition is set to be the Maxwell–Boltzmann distribution $f_{\mathrm{eq}}(\boldsymbol{v})$ in (44). A properly working algorithm should keep such equilibrium distribution unchanged as the simulation progresses. Fig. 7 shows the distribution of molecular velocities $f(\boldsymbol{v}, t)$ as function of the molecular speed $\|\boldsymbol{v}\|_2\,|\boldsymbol{\xi}|$ at various dimensionless times t/τ_R, where τ_R is the relaxation time in Table 4. It is seen that the alternating least squares algorithm we developed in Section 3.1 has a stable fixed point at the Maxwell–Boltzmann equilibrium distribution f_{eq}. This is an important test, as convergence of alternating least square is, in general, not granted for arbitrary residuals (e.g., [33,68]).

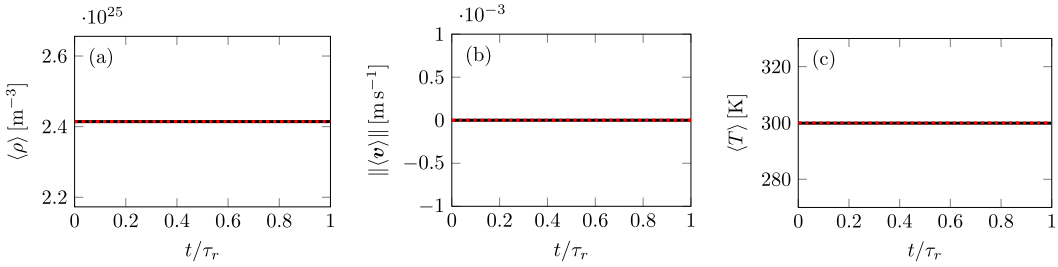

Fig. 8. Average density $\langle \rho \rangle (t)$, absolute average velocity $\|\langle u \rangle\|_2$, and average temperature $\langle T \rangle$ as function of the dimensionless time t/τ_R: simulation results (——); results at equilibrium (·····). The simulation results are obtained by setting the initial condition in (43) equal to the Maxwell–Boltzmann equilibrium distribution f_{eq} and solving the Boltzmann–BGK model with the ALS algorithm described in Section 3.1. Clearly, ALS iterations preserve the average density, velocity and temperature.

We also computed the moments of the distribution function $f(\boldsymbol{x}, \boldsymbol{v}, t)$ at different times to verify whether ALS iterations preserve (density), momentum (velocity), and kinetic energy (temperature). In particular, we evaluated

$$\langle \rho \rangle (t) = \frac{1}{b_x^3} \int_{-b_x}^{b_x} \cdots \int_{-b_v}^{b_v} f(\boldsymbol{x}, \boldsymbol{v}, t) \, \mathrm{d}\boldsymbol{x} \mathrm{d}\boldsymbol{v}, \tag{56}$$

$$\langle \boldsymbol{v} \rangle (t) = \frac{1}{\langle \rho \rangle \, b_r^3} \int_{-b_x}^{b_x} \cdots \int_{-b_v}^{b_v} \boldsymbol{v} f(\boldsymbol{x}, \boldsymbol{v}, t) \, \mathrm{d}\boldsymbol{x} \mathrm{d}\boldsymbol{v}, \tag{57}$$

$$\langle T \rangle (t) = \frac{1}{\langle \rho \rangle \, b_x^3 R_S} \int_{-b_x}^{b_x} \cdots \int_{-b_v}^{b_v} \| \boldsymbol{v} - \langle \boldsymbol{u} \rangle \|_2^2 \, f(\boldsymbol{x}, \boldsymbol{v}, t) \, \mathrm{d}\boldsymbol{x} \mathrm{d}\boldsymbol{v}. \tag{58}$$

Fig. 8 exhibits the quantities in (56)–(58) as functions of dimensionless time. These quantities are indeed constants, i.e., ALS iterations preserve the average density, velocity and temperature.

4.2.3. Relaxation to statistical equilibrium

In this section we consider relaxation to statistical equilibrium in the linearized Boltzmann–BGK model (43). This allows us to study the accuracy and computational efficiency of the proposed ALS algorithm in transient dynamics. To this end, we consider the initial condition

$$f(\boldsymbol{x}, \boldsymbol{v}, 0) = f_{eq} \left(1 + \epsilon \prod_{k=1}^{3} \cos \left(2\pi \frac{x_k}{b_x} \right) \right) \tag{59}$$

with $\epsilon = 0.3$. Such initial condition is a slight perturbation of the Maxwell–Boltzmann equilibrium distribution (Fig. 9a). The simulation results are shown in Figs. 9b–d, where we plot the time evolution of the marginal PDF $f(\boldsymbol{v}, t)$ versus the dimensionless time t/τ_R. The initial condition (59) is in the basin of attraction of f_{eq}, i.e., the ALS algorithm sends $f(\boldsymbol{x}, \boldsymbol{v}, 0)$ into $f_{\mathrm{eq}}(\boldsymbol{v})$ after a transient. Moreover, $f(\boldsymbol{v}, t)$ is attracted by $f_{\mathrm{eq}}(\boldsymbol{v})$ exponentially fast in time at a rate $\tau_R = 1/\nu$ (Fig. 11), which is consistent with results of perturbation analysis.

Fig. 10 demonstrates that our parallel ALS algorithm conserves the average density of particle, velocity (momentum) and temperature throughout the transient that yields statistical equilibrium.

Fig. 11 exhibits the Normalized Mean Absolute Error (NMAE) between the Maxwell–Boltzmann distribution $f_{\mathrm{eq}}(\boldsymbol{v})$ and the distribution we computed numerically by using the proposed ALS algorithm. Specifically, we plot NMAE versus the dimensionless time t/τ_R for the different time-steps Δt we employed in our simulations. The goal of this study is to assess the sensitivity of parallel ALS iterations on the time step Δt, and in turn on the total number of steps for a fixed integration time $T = 5\tau_R$. The results of Fig. 11 demonstrate that the decay of the distribution function $f(\boldsymbol{x}, \boldsymbol{v}, t)$ to its equilibrium state is indeed exponential. Moreover, the ALS algorithm is robust to Δt, which implies that the such algorithm is not sensitive to the total number of time steps we consider within a fixed integration time.

Finally, we study the scalability of our parallel ALS algorithm developed in Appendix B, i.e., its performance as a function of the number of degrees of freedom and the number of CPUs. To this end, we analyze a few test cases ran on a small workstation with 4 CPUs. The parallel implementation of the ALS algorithm assigns one core per dimension, which amounts to a maximum of 6 cores in the case of the Boltzmann–BGK equation. The initial condition is set to f_{eq} in all cases and the Boltzmann–BGK model (43) is integrated for 1000 steps with $\Delta t = 0.01\tau_R$. The separation rank of the solution PDF is set to $r = 1$, see functional-SSE.

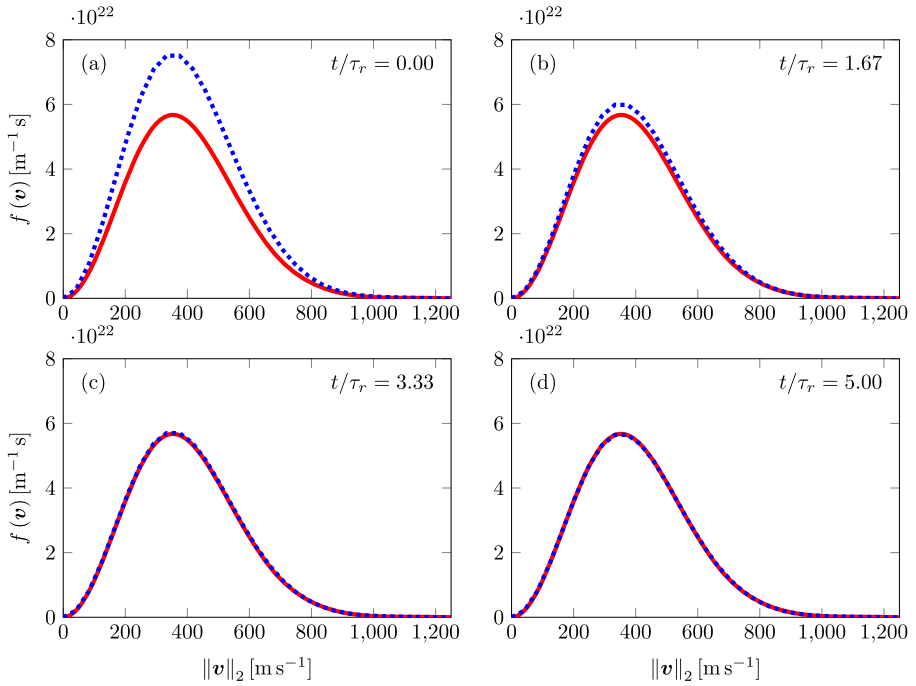

Fig. 9. Relaxation to statistical equilibrium in the linearized Boltzmann–BGK model (43). Shown are time snapshots of the marginal PDF $p(\boldsymbol{v}, t)$ we obtain by using the parallel ALS algorithm we proposed in Section 3.1 ( ). Note that the initial condition $f(\boldsymbol{v}, 0)$ is a slight perturbation of the Maxwell–Boltzmann distribution ( ). Note that $f(\boldsymbol{v}, t)$ converges to $f_{\mathrm{eq}}(\boldsymbol{v})$ exponentially fast in time (Fig. 11), consistently with results of perturbation analysis.

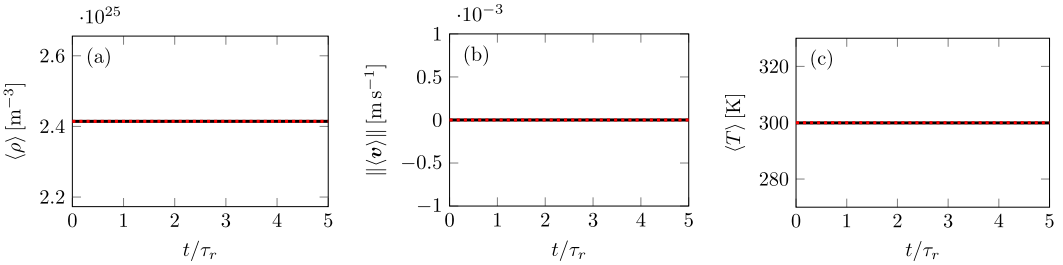

Fig. 10. Conservation of the average density $\langle\rho\rangle$, absolute average velocity $\|\langle\boldsymbol{v}\rangle\|_2$, and average temperature $\langle T\rangle$ during relaxation to statistical equilibrium. Shown are results obtained by using the parallel ALS algorithm we developed in Section 3.1 ( ), and the equilibrium initial values ( ).

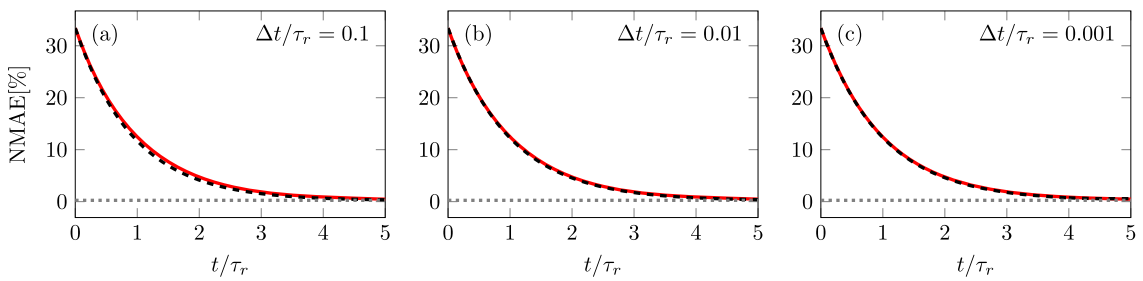

Fig. 11. Relaxation to statistical equilibrium. Normalized Mean Absolute Error (NMAE) between the Maxwell–Boltzmann distribution $f_{\mathrm{eq}}(\boldsymbol{v})$ and the numerical solution we obtain with the parallel ALS algorithm we proposed in Section 3.1 ( ). We also plot the theoretically predicted exponential decay ( ), and the best approximation error that we obtained based on approximating the equilibrium distribution f_{eq} with a canonical tensor decomposition (1) with $Q = 11$ modes ( ), $Q = 11$ being the number of modes employed in the transient simulation.

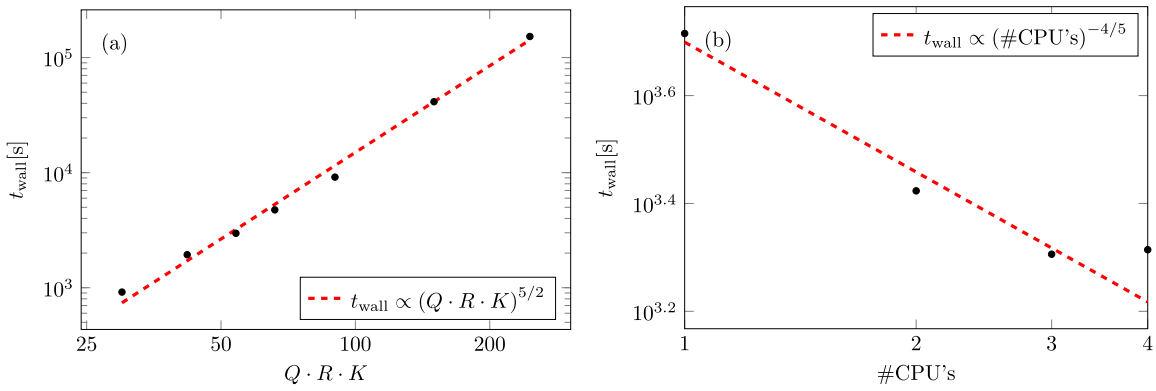

Fig. 12. Wall time t_{wall} versus the number of degrees of freedom of the ALS simulation (a), and versus the number CPUs (b).

Fig. 12a shows the wall time t_{wall} as function of the total number of degrees of freedom of the ALS algorithm, $6Qr$. A typical simulation of the six-dimensional Boltzmann–BGK model with $r = 1$ and $Q = 11$ takes about $t_{\text{wall}} \approx 80$ min on a single core to complete a simulation up $t = 10$ s. With such value of Q there is an error of only 0.5% compared to the analytical Maxwell–Boltzmann distribution function. The total wall time scales with the power 5/2 relative to number of degrees of freedom $6Qr$. To put this number into perspective, the worst case scenario for matrix inversion, the system scales as $\mathcal{O}(n^3)$. On the other hand, the complete decoupling of the different dimensions results in a scaling of $\mathcal{O}(n)$ with respect to dimension size. Both of these limits show a growth rate that is slower than the exponential scaling typically found in systems suffering from the curse of dimensionality.

Fig. 12b demonstrates that the total wall time scales almost linearly with the number of CPUs (power $-4/5$). The leveling off of performance for 4 cores can be attributed to the fact that the desktop computer we employed for our simulations is optimized for inter-core communication. We expect better scaling performance of our ALS algorithm in computer clusters with InfinyBand inter-core communication.

5. Summary

In this paper we presented new parallel algorithms to solve high-dimensional partial differential equations based on numerical tensor methods. In particular, we studied canonical and hierarchical tensor expansions combined with alternating least squares and high-order singular value decomposition. Both implicit and explicit time integration methods were presented and discussed. We demonstrated the accuracy and computational efficiency of the proposed methods in simulating the transient dynamics generated by a linear advection equation in six dimensions plus time, and the well-known Boltzmann–BGK equation. We found that the algorithms we developed are extremely fast and accurate, even in a naive Matlab implementation. Unlike direct simulation Monte Carlo (DSMC), the high accuracy of the numerical tensor methods we propose makes it suitable for studying transient problems. On the other hand, given the nature of the numerical tensor discretization, implementation of the proposed algorithms to complex domains is not straightforward.

Acknowledgements

We would like to thank N. Urien for useful discussions. D. Venturi was supported by the AFOSR grant FA9550-16-1-0092. Arnout Boelens and Daniel Tartakovsky were supported in part by the NSF grant DMS-1802189 awarded to the latter.

Appendix A. Boltzmann equation and its approximations

In the classical kinetic theory of rarefied gas dynamics, the gas flow is described in terms of a non-negative density function $f(\boldsymbol{x}, \boldsymbol{v}, t)$ which provides the number of gas particles at time t with velocity $\boldsymbol{v} \in \mathbb{R}^3$ at position $\boldsymbol{x} \in \mathbb{R}^3$. The density function f satisfies the Boltzmann equation [69,70,63]. In the absence of external forces, such equation can be written as

$$\frac{\partial f}{\partial t} + \boldsymbol{v} \cdot \nabla_{\boldsymbol{x}} f = Q(f, f), \tag{A.1}$$

where $Q(f, f)$ is the collision integral describing the effects of internal forces due to particle interactions. From a mathematical viewpoint the collision integral is a functional of the particle density. Its form depends on the microscopic dynamics. For example, for classical rarefied gas flows [69,71],

$$Q(f, f)(\boldsymbol{x}, \boldsymbol{v}, t) = \int_{\mathbb{R}^3} \int_{\mathbb{S}^2} B(\boldsymbol{v}, \boldsymbol{v}_1, \boldsymbol{\omega}) \left| f(\boldsymbol{x}, \boldsymbol{v}', t) f(\boldsymbol{x}, \boldsymbol{v}_1', t) - f(\boldsymbol{x}, \boldsymbol{v}, t) f(\boldsymbol{x}, \boldsymbol{v}_1, t) \right| \mathrm{d}\boldsymbol{\omega} \mathrm{d}\boldsymbol{v}_1. \tag{A.2}$$

In this expression,

$$\boldsymbol{v}' = \frac{1}{2}\left(\boldsymbol{v} + \boldsymbol{v}_1 + \|\boldsymbol{v} - \boldsymbol{v}_1\|_2\, \boldsymbol{w}\right), \qquad \boldsymbol{v}_1' = \frac{1}{2}\left(\boldsymbol{v} + \boldsymbol{v}_1 - \|\boldsymbol{v} - \boldsymbol{v}_1\|_2\, \boldsymbol{w}\right) \tag{A.3}$$

and $\{\boldsymbol{v}, \boldsymbol{v}_1\}$ represent, respectively, the velocities of two particles before and after the collision, $\boldsymbol{\omega}$ is a vector on the unit sphere $\mathbb{S}^2$, and $B(\boldsymbol{v}, \boldsymbol{v}_1, \boldsymbol{\omega})$ is the collision kernel. Such kernel is a non-negative function depending on the (Euclidean) norm $\|v - v_1\|_2$ and on the scattering angle θ between the relative velocities before and after the collision

$$\cos(\theta) = \frac{(\boldsymbol{v} - \boldsymbol{v}_1)\cdot\boldsymbol{\omega}}{\|\boldsymbol{v} - \boldsymbol{v}_1\|_2}. \tag{A.4}$$

Thus, $B(\boldsymbol{v}, \boldsymbol{v}_1, \boldsymbol{\omega})$ becomes

$$B(\boldsymbol{v}, \boldsymbol{v}_1, \boldsymbol{\omega}) = \|\boldsymbol{v} - \boldsymbol{v}_1\|_2\, \sigma\left(\|\boldsymbol{v} - \boldsymbol{v}_1\|_2, \cos(\theta)\right), \tag{A.5}$$

σ being the scattering cross section function [71]. The collision operator (A.2) satisfies a system of conservation laws in the form

$$\int_{\mathbb{R}^3} Q(f, f)(\boldsymbol{x}, \boldsymbol{v}, t)\psi(\boldsymbol{v})\mathrm{d}\boldsymbol{v} = 0, \tag{A.6}$$

where $\psi(\boldsymbol{v})$ is either $\{1, \boldsymbol{v}, \|\boldsymbol{v}\|_2^2\}$. This yields, respectively, conservation of mass, momentum and energy. Moreover, $Q(f, f)$ satisfies the Boltzmann H-theorem

$$\int_{\mathbb{R}^3} Q(f, f)(\boldsymbol{x}, \boldsymbol{v}, t)\log\left(f(\boldsymbol{x}, \boldsymbol{v}, t)\right)\mathrm{d}\boldsymbol{v} \leq 0 \tag{A.7}$$

Such theorem implies that any equilibrium distribution function, i.e., any function f for which $Q(f, f) = 0$, has the form of a locally Maxwellian distribution

$$f_{\mathrm{eq}}(\boldsymbol{x}, \boldsymbol{v}, t) = \frac{\rho(\boldsymbol{x}, t)}{(2\pi RT(\boldsymbol{x}, t))^{3/2}} \exp\left(-\frac{\|\boldsymbol{u}(\boldsymbol{x}, t) - \boldsymbol{v}\|_2^2}{2RT(\boldsymbol{x}, t)}\right), \tag{A.8}$$

where R is the gas constant, and $\rho(\boldsymbol{x}, t)$, $\boldsymbol{u}(\boldsymbol{x}, t)$ and $T(\boldsymbol{x}, t)$ are the density, mean velocity and temperature of the gas, defined as

$$\rho(\boldsymbol{x}, t) = \int_{\mathbb{R}^3} f(\boldsymbol{x}, \boldsymbol{v}, t)\mathrm{d}\boldsymbol{v}, \tag{A.9}$$

$$\boldsymbol{u}(\boldsymbol{x}, t) = \frac{1}{\rho(\boldsymbol{x}, t)}\int_{\mathbb{R}^3} \boldsymbol{v} f(\boldsymbol{x}, \boldsymbol{v}, t)\mathrm{d}\boldsymbol{v}, \tag{A.10}$$

$$T(\boldsymbol{x}, t) = \frac{1}{3\rho(\boldsymbol{x}, t)}\int_{\mathbb{R}^3} \|\boldsymbol{u}(\boldsymbol{x}, t) - \boldsymbol{v}\|_2^2\, f(\boldsymbol{x}, \boldsymbol{v}, t)\mathrm{d}\boldsymbol{v}. \tag{A.11}$$

From a mathematical viewpoint, the Boltzmann equation (A.1) is a nonlinear integro-differential equation for a positive scalar field in six dimensions plus time. By taking suitable averages over small volumes in position space, it can be shown that the Boltzmann equation is consistent with the compressible Euler's equations [72,73], and the Navier–Stokes equations [74–76].

BGK approximation. The simplest collision operator satisfying the conservation law (A.6) and the condition

$$Q(f, f) = 0 \Leftrightarrow f(\boldsymbol{x}, \boldsymbol{v}, t) = f_{\mathrm{eq}}(\boldsymbol{x}, \boldsymbol{v}, t) \tag{A.12}$$

(see equation (A.8)) was proposed by Bhatnagar, Gross and Krook in [77]. The corresponding model, which is known as the BGK model, is defined by the linear relaxation operator

$$Q(f, f) = \nu(\boldsymbol{x}, t)\left[f_{\mathrm{eq}}(\boldsymbol{x}, \boldsymbol{v}, t) - f(\boldsymbol{x}, \boldsymbol{v}, t)\right] \qquad \nu(\boldsymbol{x}, t) > 0. \tag{A.13}$$

The field ν is usually set to be proportional to the density and the temperature of the gas [78]

$$\nu(\boldsymbol{x}, t) = \kappa\rho(\boldsymbol{x}, t)T(\boldsymbol{x}, t)^{1-\mu}. \tag{A.14}$$

It can be shown that the Boltzmann–BGK model (A.13) converges to the correct Euler equations of incompressible fluid dynamics with the scaling $\boldsymbol{x}' = \epsilon \boldsymbol{x}$, $t' = \epsilon t$, and in the limit $\epsilon \to 0$. However, the model does not converge to the correct Navier–Stokes limit. The main reason is that it predicts an unphysical Prandtl number [79], which is larger than one obtained with the full collision operator (A.2). The correct Navier–Stokes limit can be recovered by more sophisticated BGK-type models, e.g., the ellipsoidal statistical BGK model [80].

Following [81–84], we introduce additional simplifications, namely we assume that $\nu(\boldsymbol{x}, t) = \nu$ is constant and assume the equilibrium density, temperature and velocity to be homogeneous across the spatial domain. Assuming $\boldsymbol{u}(\boldsymbol{x}, t) = \boldsymbol{0}$, this yields an equilibrium distribution in (44). The last assumption effectively decouples the BGK collision operator from the probability density function $f(\boldsymbol{x}, \boldsymbol{v}, t)$. This, in turn, makes the BGK approximation linear, i.e., a six-dimensional PDE (43). Using both the isothermal assumption $T(\boldsymbol{x}, t) = T$ [77], and constant density assumption $\rho(\boldsymbol{x}, t) = \rho$ [82–84] means that our code can only be used to obtain solutions which are small perturbations from the equilibrium distribution. The term $\nu\left[f_{\text{eq}}(\boldsymbol{v}) - f\right]$ in (43) represents the relaxation time approximation of the Boltzmann equation, ν is the collision frequency, here assumed to be homogeneous.

Appendix B. Parallelization of the ALS algorithm for the Boltzmann–BGK equation

In this appendix we describe the parallel alternating least squares algorithms we developed to solve the linearized Boltzmann–BGK equation (43). Algorithm 1 shows the main Alternating Least Square (ALS) routine while Algorithm 2, 3, 4, and 5 show the different subroutines. Each time step, n, the value of β_{Old} is updated and then β_{New} is determined in an iterative manner. Both the computations of the matrices M_k and the vectors γ_k, as well as the calculation of the vectors $\beta_{\text{New},k}$ can be performed independently of each other, and thus can be easily parallelized. The current implementation assigns one core per dimension, but further optimization could be possible by implementing a parallel method to determine β_{New} [85]. At each time step the CreateMatrixM subroutine (Algorithm 2) updates the different elements of matrix M_k for

Algorithm 1 Parallel Alternating Least Squares (ALS) algorithm.

```
1: procedure MAIN
2:     INITIALIZATION                                   ▷ Load variables and allocate matrices
3:     β_New = β_0                                      ▷ Set initial condition
4:     for N ← 1 : N_Iter do
5:         ε_Conv ← ε_Tol + 1                           ▷ Reset convergence condition
6:         β_Old ← β_New
7:         while ε_Conv > ε_Tol do
8:             β_Int ← β_New
9:             parfor k ← 1 : K do                      ▷ This for loop is executed in parallel
10:                M_k ← CREATEMATRIXM(β_New,k, k)
11:                γ_k ← CREATEVECTORGAMMA(β_New,k, β_Old,k, k)
12:            parfor k ← 1 : K do                      ▷ This for loop is executed in parallel
13:                β_New,k ← COMPUTEBETA(M_k, β_New,k, γ_k)   ▷ Compute updated β_New
14:            ε_Conv ← COMPUTECONVERGENCE(β_Int, β_New)  ▷ Update convergence condition
15:            β_New ← β_Int + (β_New − β_Int)/4          ▷ Update β_New
16: end procedure
```

every iteration until convergence has been reached. The algorithm iterates over every element of the matrix and performs the multiplication and summation of the elements of $\hat{M}$. All the integrals were performed analytically and the results are stored in the map $\mathrm{M}(k, l, z, s, q)$. Using this approach delivers significant savings in the computational cost of updating the matrix every iteration. The subroutine `CreateVectorGamma` (Algorithm 3) updates the vector γ_k. Similarly to the subroutine for the creation of matrix M_k this algorithm iterates over all elements of the vector, and fills each of them using the values of β_{New} and β_{Old}, and pre-calculated maps.

With the completion of the calculation of M_k and γ_k, β_{New} can be updated using the `ComputeBeta` subroutine (Algorithm 4). Matrix M_k, depending on the rank r_G, and series truncation Q, can become very large. In addition, it is not known whether the matrix is invertible. This is why the system $M_k \beta_k = \gamma_k$ is solved using the iterative least square method, LSQR [86,87]. For every iteration the value for $\beta_{\text{New},k}$ which was determined in the previous iteration is used as the initial guess for the algorithm.

Convergence of the system is calculated using the `ComputeConvergence` subroutine (Algorithm 5). Because it is very computationally expensive to calculate the full residual every iteration, instead the convergence is determined using:

$$\epsilon_{\text{Conv}} = \max\left(\frac{\left\|\beta_{\text{New},k} - \beta_{\text{Int},k}\right\|}{\left\|\beta_{\text{Int},k}\right\|}\right) \tag{B.1}$$

and when $\epsilon_{\text{Conv}} < \epsilon_{\text{Tol}}$, the while loop in Algorithm 1 is allowed to exit. In its currently implementation the algorithm is able to reach convergence using a fixed Rank G, $r_G = 2$. However, a future implementation could implement a dynamic Rank G for improved accuracy. Every iteration the new value of β_{New} is calculated according to:

Algorithm 2 ALS Subroutines: CreateMatrixM.

function CreateMatrixM($\beta_{\text{New},k}, k$)
  for $l \leftarrow 1 : R_G$ **do** ▷ Loop over all elements of M_k
    for $z \leftarrow 1 : R_G$ **do**
      for $s \leftarrow -|Q|/2 : |Q|/2$ **do**
        for $q \leftarrow -|Q|/2 : |Q|/2$ **do**
          for $k' \leftarrow 1 : K$ **do** ▷ Loop over all elements of $\hat{M}$ for each element of M_k
            for $\eta \leftarrow 1 : r_A$ **do**
              for $\upsilon \leftarrow 1 : r_A$ **do**
                $j \leftarrow (\eta - 1) r_A + \upsilon$
                if k' = k **then**
                  $\hat{M}_{k',j} \leftarrow \mathrm{M}(k', \eta, \upsilon, s, q)$
                else
                  $\hat{M}_{k',j} \leftarrow 0$
                  for $s' \leftarrow -|Q|/2 : |Q|/2$ **do**
                    for $q' \leftarrow -|Q|/2 : |Q|/2$ **do**
                      $\hat{M}_{k',j} \leftarrow \hat{M}_{k',j} + \beta_{\text{New},k',(l-1)Q+s'} \beta_{\text{New},k',(z-1)Q+q'} \mathrm{M}(k', \eta, \upsilon, s', q')$
          $M_{k,(z-1)Q+q,(l-1)Q+s} \leftarrow \sum_{j=1}^{r_A^2} \prod_{k'=1}^{K} \hat{M}_{k',j}$ ▷ Fill matrix M_k
  return M_k
end function

Algorithm 3 ALS Subroutines: CreateVectorGamma.

function CreateVectorGamma($\beta_{\text{New},k}, \beta_{\text{Old},k}, k$)
  for $z \leftarrow 1 : R_G$ **do** ▷ Loop over all elements of γ_k
    for $q \leftarrow -|Q|/2 : |Q|/2$ **do**
      for $k' \leftarrow 1 : K$ **do** ▷ Loop over all elements of $\hat{N}$ for each element of γ_k
        for $l' \leftarrow 1 : R_G$ **do**
          for $\eta \leftarrow 1 : r_A$ **do**
            for $\upsilon \leftarrow 1 : r_A$ **do**
              $j \leftarrow (\eta - 1) r_A + \upsilon$
              if k' = k **then**
                $\hat{N}_{k',l',j} \leftarrow 0$
                for $s' \leftarrow -|Q|/2 : |Q|/2Q$ **do**
                  $\hat{N}_{k',l',j} \leftarrow \hat{N}_{k',l',j} + \beta_{\text{Old},k',(l'-1)Q+s'} \mathrm{N}(k', \eta, \upsilon, s', q)$
              else
                $\hat{N}_{k',l',j} \leftarrow 0$
                for $s' \leftarrow -|Q|/2 : |Q|/2Q$ **do**
                  for $q' \leftarrow -|Q|/2 : |Q|/2$ **do**
                    $\hat{N}_{k',l',j} \leftarrow \hat{N}_{k',l',j} + \beta_{\text{Old},k',(l'-1)Q+s'} \beta_{\text{New},k',(z-1)Q+q'} \mathrm{N}(k', \eta, \upsilon, s', q')$
      $N_{k,(z-1)Q+q} \leftarrow \sum_{l'=1}^{R_G} \sum_{j=1}^{r_A^2} \prod_{k'=1}^{K} \hat{N}_{k',l',j}$ ▷ Fill matrix N_k
      for $k' \leftarrow 1 : K$ **do** ▷ Loop over all elements of $\hat{O}$ for each element of γ_k
        for $\eta \leftarrow 1 : r_A$ **do**
          if k' = k **then**
            $\hat{O}_{k',\eta} \leftarrow \mathrm{O}(k', \eta, q)$
          else
            $\hat{O}_{k',\eta} \leftarrow 0$
            for $q' \leftarrow -|Q|/2 : |Q|/2$ **do**
              $\hat{O}_{k',\eta} \leftarrow \hat{O}_{k',\eta} + \beta_{\text{New},k',(z-1)Q+q'} \mathrm{O}(k', \eta, q')$
      $O_{k,(z-1)Q+q} \leftarrow \sum_{\eta=1}^{r_A} \prod_{k'=1}^{K} \hat{O}_{k',\eta}$ ▷ Fill matrix O_k
  $\gamma_k = N_k + \Delta t \; \upsilon \; O_k$
  return γ_k
end function

Algorithm 4 ALS Subroutines: ComputeBeta.

function ComputeBeta($M_k, \beta_{\text{New},k}, \gamma_k$)
  $\beta_{\text{New},k} \leftarrow \mathrm{Lsqr}(M_k, \beta_{\text{New},k}, \gamma_k)$ ▷ The system $\gamma_k - M_k \beta_{\text{New},k} = \text{Res}$ is solved using the least squares method. $\beta_{\text{New},k}$ is used as an initial guess before being updated
  return $\beta_{\text{New},k}$
end function

$$\beta_{\text{New}} \leftarrow \beta_{\text{Int}} + \frac{\beta_{\text{New}} - \beta_{\text{Int}}}{\Delta\beta} \tag{B.2}$$

with $\Delta\beta = 4$. To find the location where the residual reaches a minimum, it is important to gradually approach this location and not update β_{New} too aggressively by taking a smaller value of $\Delta\beta$. This causes overshooting of the minimum and the prevents convergence of β_{New} and thus minimization of the residual.

Algorithm 5 ALS Subroutines: ComputeConvergence.

function ComputeConvergence($\beta_{\text{Int}}, \beta_{\text{New}}$)
  for $k \leftarrow 1 : K$ **do**
    $\beta_{\text{Norm},k} \leftarrow \|\beta_{\text{New},k} - \beta_{\text{Int},k}\| / \|\beta_{\text{Int},k}\|$
  return $\max(\beta_{\text{Norm}})$
end function

References

[1] C. Cercignani, The Boltzmann Equation and Its Applications, Springer, New York, 1988.
[2] F. Moss, P.V.E. McClintock (Eds.), Noise in Nonlinear Dynamical Systems, Cambridge University Press, 1989.
[3] D. Venturi, T.P. Sapsis, H. Cho, G.E. Karniadakis, A computable evolution equation for the joint response-excitation probability density function of stochastic dynamical systems, Proc. R. Soc. A 468 (2139) (2011) 759–783, https://doi.org/10.1098/rspa.2011.0186.
[4] M.F. Shlesinger, T. Swean, Stochastically Excited Nonlinear Ocean Structures, World Scientific, 1998.
[5] A.S. Monin, A.M. Yaglom, Statistical Fluid Mechanics, Volume II: Mechanics of Turbulence, Dover, 2007.
[6] U. Frisch, Turbulence: The Legacy of A.N. Kolmogorov, Cambridge University Press, 1995.
[7] D. Venturi, The numerical approximation of nonlinear functionals and functional differential equations, Phys. Rep. 732 (2018) 1–102, https://doi.org/10.1016/j.physrep.2017.12.003.
[8] L. Cowen, T. Ideker, B.J. Raphael, R. Sharan, Network propagation: a universal amplifier of genetic associations, Nat. Rev. Genet. 18 (9) (2017) 551–562, https://doi.org/10.1038/nrg.2017.38.
[9] J. Zinn-Justin, Quantum Field Theory and Critical Phenomena, 4th edition, Oxford University Press, 2002.
[10] D.J. Amit, V. Martin-Mayor, Field Theory, the Renormalization Group, and Critical Phenomena, World Scientific, 2005.
[11] H. Risken, The Fokker–Planck Equation, 2nd edition, Springer Berlin Heidelberg, 1989.
[12] D.M. Tartakovsky, P.A. Gremaud, Method of distributions for uncertainty quantification, in: R. Ghanem, D. Higdon, H. Owhadi (Eds.), Handbook of Uncertainty Quantification, Springer International Publishing, Switzerland, 2017, pp. 763–783.
[13] S.B. Pope, Lagrangian PDF methods for turbulent flows, Annu. Rev. Fluid Mech. 26 (1) (1994) 23–63, https://doi.org/10.1146/annurev.fl.26.010194.000323.
[14] G. Dimarco, L. Pareschi, Numerical methods for kinetic equations, Acta Numer. 23 (2014) 369–520, https://doi.org/10.1017/s0962492914000063.
[15] S. Rjasanow, W. Wagner, Stochastic Numerics for the Boltzmann Equation, Springer, 2004.
[16] H. Babovsky, H. Neunzert, On a simulation scheme for the Boltzmann equation, Math. Methods Appl. Sci. 8 (1) (1986) 223–233, https://doi.org/10.1002/mma.1670080114.
[17] S. Ansumali, Mean-field model beyond Boltzmann–Enskog picture for dense gases, Commun. Comput. Phys. 9 (05) (2011) 1106–1116, https://doi.org/10.4208/cicp.301009.240910s.
[18] S. Ramanathan, D.L. Koch, An efficient direct simulation Monte Carlo method for low Mach number noncontinuum gas flows based on the Bhatnagar–Bross–Krook model, Phys. Fluids 21 (3) (2009) 033103, https://doi.org/10.1063/1.3081562.
[19] J.-P.M. Peraud, C.D. Landon, N.G. Hadjiconstantinou, Monte Carlo methods for solving the Boltzmann transport equation, Annu. Rev. Heat Transf. 17 (2014) 205–265, https://doi.org/10.1615/annualrevheattransfer.2014007381.
[20] F. Rogier, J. Schneider, A direct method for solving the Boltzmann equation, Transp. Theory Stat. Phys. 23 (1–3) (1994) 313–338, https://doi.org/10.1080/00411459408203868.
[21] H. Cho, D. Venturi, G. Karniadakis, Numerical methods for high-dimensional probability density function equations, J. Comput. Phys. 305 (2016) 817–837, https://doi.org/10.1016/j.jcp.2015.10.030.
[22] Z. Zhang, G.E. Karniadakis, Numerical Methods for Stochastic Partial Differential Equations with White Noise, Springer International Publishing, 2017.
[23] W. E, J. Han, A. Jentzen, Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations, Commun. Math. Stat. 5 (4) (2017) 349–380, https://doi.org/10.1007/s40304-017-0117-6.
[24] M. Bachmayr, R. Schneider, A. Uschmajew, Tensor networks and hierarchical tensors for the solution of high-dimensional partial differential equations, Found. Comput. Math. 16 (6) (2016) 1423–1472, https://doi.org/10.1007/s10208-016-9317-9.
[25] J.S. Hesthaven, S. Gottlieb, D. Gottlieb, Spectral Methods for Time-Dependent Problems, vol. 21, Cambridge University Press, 2007.
[26] G. Beylkin, J. Garcke, M.J. Mohlenkamp, Multivariate regression and machine learning with sums of separable functions, SIAM J. Sci. Comput. 31 (3) (2009) 1840–1857, https://doi.org/10.1137/070710524.
[27] E. Acar, D.M. Dunlavy, T.G. Kolda, A scalable optimization approach for fitting canonical tensor decompositions, J. Chemom. 25 (2) (2011) 67–86, https://doi.org/10.1002/cem.1335.
[28] M. Espig, W. Hackbusch, A regularized Newton method for the efficient approximation of tensors represented in the canonical tensor format, Numer. Math. 122 (3) (2012) 489–525, https://doi.org/10.1007/s00211-012-0465-9.
[29] L. Karlsson, D. Kressner, A. Uschmajew, Parallel algorithms for tensor completion in the CP format, Parallel Comput. 57 (2016) 222–234, https://doi.org/10.1016/j.parco.2015.10.002.
[30] A. Doostan, A. Validi, G. Iaccarino, Non-intrusive low-rank separated approximation of high-dimensional stochastic models, Comput. Methods Appl. Mech. Eng. 263 (2013) 42–55, https://doi.org/10.1016/j.cma.2013.04.003.
[31] M.J. Reynolds, A. Doostan, G. Beylkin, Randomized alternating least squares for canonical tensor decompositions: application to a PDE with random data, SIAM J. Sci. Comput. 38 (5) (2016) A2634–A2664, https://doi.org/10.1137/15m1042802.
[32] C. Battaglino, G. Ballard, T.G. Kolda, A practical randomized CP tensor decomposition, arXiv:1701.06600, 2017.
[33] A. Uschmajew, Local convergence of the alternating least squares algorithm for canonical tensor approximation, SIAM J. Matrix Anal. Appl. 33 (2) (2012) 639–652, https://doi.org/10.1137/110843587.
[34] M. Espig, W. Hackbusch, A. Khachatryan, On the convergence of alternating least squares optimisation in tensor format representations, arXiv:1506.00062, 2015.
[35] J.C. Bezdek, R.J. Hathaway, Convergence of alternating optimization, Neural Parallel Sci. Comput. 11 (2003) 351–368.
[36] T. Rohwedder, A. Uschmajew, On local convergence of alternating schemes for optimization of convex problems in the tensor train format, SIAM J. Numer. Anal. 51 (2) (2013) 1134–1162, https://doi.org/10.1137/110857520.
[37] J.M. Ortega, W.C. Rheinboldt, Iterative Solution of Nonlinear Equations in Several Variables, Society for Industrial and Applied Mathematics, 2000.
[38] O. Kaya, B. Ucar, Parallel Candecomp/Parafac decomposition of sparse tensors using dimension trees, SIAM J. Sci. Comput. 40 (1) (2018) C99–C130, https://doi.org/10.1137/16m1102744.
[39] W. Hackbusch, S. Kühn, A new scheme for the tensor representation, J. Fourier Anal. Appl. 15 (5) (2009) 706–722, https://doi.org/10.1007/s00041-009-9094-9.
[40] W. Hackbusch, Tensor Spaces and Numerical Tensor Calculus, Springer Berlin Heidelberg, 2012.

[41] T.G. Kolda, B.W. Bader, Tensor decompositions and applications, SIAM Rev. 51 (3) (2009) 455–500, https://doi.org/10.1137/07070111x.
[42] L. Grasedyck, Hierarchical singular value decomposition of tensors, SIAM J. Matrix Anal. Appl. 31 (4) (2010) 2029–2054, https://doi.org/10.1137/090764189.
[43] L. De Lathauwer, B. De Moor, J. Vandewalle, A multilinear singular value decomposition, SIAM J. Matrix Anal. Appl. 21 (4) (2000) 1253–1278, https://doi.org/10.1137/s0895479896305696.
[44] D. Venturi, A fully symmetric nonlinear biorthogonal decomposition theory for random fields, Physica D 240 (4–5) (2011) 415–425, https://doi.org/10.1016/j.physd.2010.10.005.
[45] A. Peres, Higher order Schmidt decompositions, Phys. Lett. A 202 (1) (1995) 16–17, https://doi.org/10.1016/0375-9601(95)00315-t.
[46] V. de Silva, L.-H. Lim, Tensor rank and the ill-posedness of the best low-rank approximation problem, SIAM J. Matrix Anal. Appl. 30 (3) (2008) 1084–1127, https://doi.org/10.1137/06066518x.
[47] C.J. Hillar, L.-H. Lim, Most tensor problems are NP-hard, J. ACM 60 (6) (2013) 1–39, https://doi.org/10.1145/2512329.
[48] N. Vannieuwenhoven, J. Nicaise, R. Vandebril, K. Meerbergen, On generic nonexistence of the Schmidt–Eckart–Young decomposition for complex tensors, SIAM J. Matrix Anal. Appl. 35 (3) (2014) 886–903, https://doi.org/10.1137/130926171.
[49] D. Kressner, C. Tobler, Algorithm 941, ACM Trans. Math. Softw. 40 (3) (2014) 1–22, https://doi.org/10.1145/2538688.
[50] C. Da Silva, F.J. Herrmann, Optimization on the hierarchical Tucker manifold – applications to tensor completion, Linear Algebra Appl. 481 (2015) 131–173, https://doi.org/10.1016/j.laa.2015.04.015.
[51] I.V. Oseledets, Tensor-train decomposition, SIAM J. Sci. Comput. 33 (5) (2011) 2295–2317, https://doi.org/10.1137/090752286.
[52] L. Grasedyck, R. Kriemann, C. Löbbert, A. Nägel, G. Wittum, K. Xylouris, Parallel tensor sampling in the hierarchical Tucker format, Comput. Vis. Sci. 17 (2) (2015) 67–78, https://doi.org/10.1007/s00791-015-0247-x.
[53] A. Nouy, Higher-order principal component analysis for the approximation of tensors in tree-based low-rank formats, arXiv:1705.00880, 2017.
[54] S. Etter, Parallel ALS algorithm for solving linear systems in the hierarchical Tucker representation, SIAM J. Sci. Comput. 38 (4) (2016) A2585–A2609, https://doi.org/10.1137/15m1038852.
[55] L. Grasedyck, C. Löbbert, Distributed Hierarchical SVD in the Hierarchical Tucker format, arXiv:1708.03340, 2017.
[56] A.-H. Phan, P. Tichavsky, A. Cichocki, CANDECOMP/PARAFAC decomposition of high-order tensors through tensor reshaping, IEEE Trans. Signal Process. 61 (19) (2013) 4847–4860, https://doi.org/10.1109/tsp.2013.2269046.
[57] G. Zhou, A. Cichocki, S. Xie, Accelerated canonical polyadic decomposition using mode reduction, IEEE Trans. Neural Netw. Learn. Syst. 24 (12) (2013) 2051–2062, https://doi.org/10.1109/tnnls.2013.2271507.
[58] D. Hatch, D. del Castillo-Negrete, P. Terry, Analysis and compression of six-dimensional gyrokinetic datasets using higher order singular value decomposition, J. Comput. Phys. 231 (11) (2012) 4234–4256, https://doi.org/10.1016/j.jcp.2012.02.007.
[59] K. Kormann, A semi-Lagrangian Vlasov solver in tensor train format, SIAM J. Sci. Comput. 37 (4) (2015) B613–B632, https://doi.org/10.1137/140971270.
[60] S. Dolgov, A. Smirnov, E. Tyrtyshnikov, Low-rank approximation in the numerical modeling of the Farley–Buneman instability in ionospheric plasma, J. Comput. Phys. 263 (2014) 268–282, https://doi.org/10.1016/j.jcp.2014.01.029.
[61] A. Quarteroni, R. Sacco, F. Saleri, Numerical Mathematics, 2nd edition, Springer, New York, 2007.
[62] H.-K. Rhee, R. Aris, N.R. Amundson, First-Order Partial Differential Equations, Volume 1: Theory and Applications of Single Equations, Dover, 2001.
[63] G. Dimarco, R. Loubère, J. Narski, T. Rey, An efficient numerical method for solving the Boltzmann equation in multidimensions, J. Comput. Phys. 353 (2018) 46–81, https://doi.org/10.1016/j.jcp.2017.10.010.
[64] L. Desvillettes, S. Mischler, About the splitting algorithm for Boltzmann and B.G.K. equations, Math. Models Methods Appl. Sci. 06 (08) (1996) 1079–1101, https://doi.org/10.1142/s0218202596000444.
[65] P. Shuleshko, A new method of solving boundary-value problems of mathematical physics, Aust. J. Appl. Sci. 10 (1959) 1–7.
[66] L.J. Snyder, T.W. Spriggs, W.E. Stewart, Solution of the equations of change by Galerkin's method, AIChE J. 10 (4) (1964) 535–540, https://doi.org/10.1002/aic.690100423.
[67] B. Zinn, E. Powell, Application of the Galerkin method in the solution of combustion-instability problems, in: XIXth International Astronautical Congress, vol. 3, 1968, pp. 59–73.
[68] P. Comon, X. Luciani, A.L.F. de Almeida, Tensor decompositions, alternating least squares and other tales, J. Chemom. 23 (7–8) (2009) 393–405, https://doi.org/10.1002/cem.1236.
[69] C. Cercignani, V.I. Gerasimenko, D.Y. Petrina, Many-Particle Dynamics and Kinetic Equations, 1st edition, Springer, Netherlands, 1997.
[70] C. Villani, A review of mathematical topics in collisional kinetic theory, in: S. Friedlander, D. Serre (Eds.), Handbook of Mathematical Fluid Mechanics, vol. 1, North-Holland, 2002, pp. 71–305.
[71] C. Cercignani, R. Illner, M. Pulvirenti, The Mathematical Theory of Dilute Gases, Springer, New York, 1994.
[72] T. Nishida, Fluid dynamical limit of the nonlinear Boltzmann equation to the level of the compressible Euler equation, Commun. Math. Phys. 61 (2) (1978) 119–148, https://doi.org/10.1007/bf01609490.
[73] R.E. Caflisch, The fluid dynamic limit of the nonlinear Boltzmann equation, Commun. Pure Appl. Math. 33 (5) (1980) 651–666, https://doi.org/10.1002/cpa.3160330506.
[74] H. Struchtrup, Macroscopic Transport Equations for Rarefied Gas Flows, Springer Berlin Heidelberg, 2005.
[75] C.D. Levermore, Moment closure hierarchies for kinetic theories, J. Stat. Phys. 83 (5–6) (1996) 1021–1065, https://doi.org/10.1007/bf02179552.
[76] I. Müller, T. Ruggeri, Rational Extended Thermodynamics, Springer, New York, 1998.
[77] 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 (3) (1954) 511–525, https://doi.org/10.1103/physrev.94.511.
[78] L. Mieussens, Discrete-velocity models and numerical schemes for the Boltzmann-BGK equation in plane and axisymmetric geometries, J. Comput. Phys. 162 (2) (2000) 429–466, https://doi.org/10.1006/jcph.2000.6548.
[79] J. Nassios, J.E. Sader, High frequency oscillatory flows in a slightly rarefied gas according to the Boltzmann–BGK equation, J. Fluid Mech. 729 (2013) 1–46, https://doi.org/10.1017/jfm.2013.281.
[80] P. Andries, P. Le Tallec, J.-P. Perlat, B. Perthame, The Gaussian–BGK model of Boltzmann equation with small Prandtl number, Eur. J. Mech. B, Fluids 19 (6) (2000) 813–830, https://doi.org/10.1016/s0997-7546(00)01103-1.
[81] N. Borghini, Topics in Nonequilibrium Physics, Sep. 2016.
[82] L. Barichello, C. Siewert, Some comments on modeling the linearized Boltzmann equation, J. Quant. Spectrosc. Radiat. Transf. 77 (1) (2003) 43–59, https://doi.org/10.1016/s0022-4073(02)00074-2.
[83] C.L. Pekeris, Z. Alterman, Solution of the Boltzmann–Hilbert integral equation. II. The coefficients of viscosity and heat conduction, Proc. Natl. Acad. Sci. 43 (11) (1957) 998–1007, https://doi.org/10.1073/pnas.43.11.998.
[84] S.K. Loyalka, Model dependence of the slip coefficient, Phys. Fluids 10 (8) (1967) 1833, https://doi.org/10.1063/1.1762366.
[85] H. Huang, J.M. Dennis, L. Wang, P. Chen, A scalable parallel LSQR algorithm for solving large-scale linear system for tomographic problems: a case study in seismic tomography, Proc. Comput. Sci. 18 (2013) 581–590, https://doi.org/10.1016/j.procs.2013.05.222.
[86] C.C. Paige, M.A. Saunders, LSQR: an algorithm for sparse linear equations and sparse least squares, ACM Trans. Math. Softw. 8 (1) (1982) 43–71, https://doi.org/10.1145/355984.355989.
[87] R. Barrett, M. Berry, T.F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, H. van der Vorst, Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, Society for Industrial and Applied Mathematics, 1994.