Nonnegative Low-Rank Matrix Correction under an Orthogonality Constraint in Conservative Vlasov Simulations

Yue Wu*, Stephen Becker†, Jingmei Qiu‡, Xiangxiong Zhang§

July 28, 2026

Abstract

In low-rank numerical methods for Vlasov dynamics, the SVD-type truncation procedure may introduce negative entries into the numerical solution. Such negative values are unphysical because the solution is a probability distribution function. We design optimization-based post-processing algorithms to recover nonnegativity while preserving the macroscopic quantities (density, momentum, and energy) pointwise. The preservation of the macroscopic quantities is written as an orthogonality constraint on the correction term. For a convex formulation based on squared nuclear norm minimization, we show that the proximal operator with the orthogonality constraint is characterized by an implicit singular value thresholding equation, and the threshold can be computed efficiently by bisection. Based on this result, we develop five algorithms for the convex formulation: Douglas–Rachford splitting, restarted dual FISTA, restarted dual accelerated gradient descent, dual PR+ conjugate gradient, and dual L-BFGS. We also consider a non-convex formulation with an explicit rank constraint and develop a tangent-space accelerated alternating projection algorithm that only requires a $2r \times 2r$ SVD per iteration. Numerical results for a Landau damping test case show that the proposed algorithms give comparable correction quality. Among them, the tangent-space accelerated alternating projection is the most cost-efficient, increasingly so as the problem size grows. We further demonstrate the correction as a positivity limiter inside a time-dependent conservative low-rank Vlasov solver, where it removes the negativity introduced by the SVD-type truncation while preserving the conserved mass, momentum, and energy.

Keywords: Nonnegative low-rank matrix approximation · Nuclear norm minimization · Orthogonality constraint · Convex optimization · Vlasov dynamics · Alternating projection on manifolds

Mathematics Subject Classification (2020) 65F55 · 15A83 · 90C25 · 90C26 · 65K10 · 35Q83

*Division of Applied Mathematics, Brown University, 182 George Street, Providence, RI 02912 (yue_wu3@brown.edu).

†Department of Applied Mathematics, University of Colorado Boulder, 526 UCB, Boulder, CO 80309 (stephen.becker@colorado.edu).

‡Department of Mathematical Sciences, University of Delaware, 501 Ewing Hall, Newark, DE 19716 (jingqiu@udel.edu).

§Corresponding author. Department of Mathematics, Purdue University, 150 N. University Street, West Lafayette, IN 47907 (zhan1966@purdue.edu). This work is partially supported by NSF DMS-2208518.

1 Introduction

Numerical simulation of the Vlasov–Poisson (VP) system is fundamental to understanding the complex dynamics of collisionless plasmas and has a wide range of applications in science and engineering, such as fusion energy and space weather prediction. The VP system describes the evolution of the probability distribution function $f(\boldsymbol{x}, \boldsymbol{v}, t)$ in phase space, which by definition must be nonnegative. Its velocity moments, including mass density, momentum density, and energy density, satisfy a system of macroscopic conservation laws. Preserving both nonnegativity and conservation at the discrete level is essential for producing physically meaningful solutions and for the nonlinear stability of the numerical scheme, especially in demanding regimes such as low density and pressure.

The high dimensionality of phase space poses a major computational challenge for deterministic VP solvers. Recently, low-rank and adaptive-rank tensor methods [9] have emerged as promising approaches for mitigating the curse of dimensionality by adaptively exploiting low-rank structure in the solution. Broadly, these methods can be divided into two classes. The first is the dynamical low-rank approach, which evolves the low-rank factors by projecting the governing equation onto the tangent space of the low-rank manifold [16]. Robust and rank-adaptive time integrators have been developed within this framework [20, 5, 4, 7]. The second is the step-and-truncate approach, in which the solution is first advanced in an enriched approximation space and subsequently compressed to control the rank. Within a Galerkin framework, the method proposed in [12] dynamically constructs an adaptive low-rank solution basis by adding new basis functions generated from the discretized PDE and removing redundant components through SVD-type truncation. Within a collocation framework [33, 34], selected rows and columns of the solution matrix are adaptively identified, allowing the solution to be updated through CUR-type matrix interpolation without forming the full high-dimensional solution. An important challenge for both classes of low-rank methods is the preservation of the physical conservation laws. Conservative SVD truncation procedures were developed in [8, 13] to preserve local mass and momentum. Recently, the Local Macroscopic Conservative (LoMaC) low-rank tensor methods developed in [14, 11] preserve mass, momentum, and energy locally at the discrete level. Their central idea is to evolve the macroscopic conservation laws alongside the kinetic equation, construct a reference subspace from the resulting macroscopic densities, and project the low-rank kinetic solution onto this subspace through a conservative orthogonal projection.

A remaining issue for the low-rank methods in [13, 14] is the lack of nonnegativity in the numerical solution. Since the distribution function f represents a probability density, it must be nonnegative everywhere. However, SVD-type low-rank approximations of a nonnegative matrix can introduce negative entries, and this problem is further compounded by the time-stepping procedure. Within the low-rank framework, positivity can be enforced by evolving a square-root factorization $f = g^2$, so that the represented solution is nonnegative by construction [31]. This reformulation is tied to the structure of the transport operator and does not extend to second-order operators such as a Fokker–Planck collision term, and a nonlinear transformation of a low-rank function need not remain low rank. We instead leave the low-rank solver unchanged and recover nonnegativity by a minimal post-processing correction, which is agnostic to the underlying operator. Enforcing nonnegativity (or more generally, preserving the invariant domain of the underlying PDE) in high-order numerical methods has been an active area of research. We refer to [30] for a comprehensive survey. The main approaches include polynomial limiters based on convex decomposition of cell averages [24, 32], flux-corrected transport (FCT) methods, and more recently, optimization-based limiters. In

the optimization-based approach, the limiter seeks a minimal modification of the numerical solution while enforcing conservation and positivity (or bounds) as constraints. Such optimization-based methods have been developed for enforcing nonnegativity of polynomial approximations [6] and for invariant-domain-preserving limiters in gas dynamics [18], where first-order operator splitting methods such as the Douglas–Rachford splitting [17] are employed to efficiently solve the resulting constrained minimization problems. At the representation level, [28] proposes a nonnegative tensor train optimization framework for high-dimensional distribution tensors, whose complexity scales linearly with both the dimension d and the number of degrees of freedom per dimension n.

In this work, we design optimization-based post-processing algorithms to recover the nonnegativity of the low-rank solution produced by [13, 14], while preserving the macroscopic quantities (density, momentum, and energy) pointwise. The problem is to find a correction term $\boldsymbol{X}$ such that $\boldsymbol{A} + \boldsymbol{X} \geqslant \boldsymbol{0}$ and $\boldsymbol{X}\boldsymbol{B} = \boldsymbol{0}$, where $\boldsymbol{A}$ is the possibly negative scaled solution matrix and $\boldsymbol{B}$ encodes the orthogonality constraint associated with the first three velocity moments. We further split $\boldsymbol{A} = \boldsymbol{A}_1 + \boldsymbol{A}_2$, where $\boldsymbol{A}_1$ is a rank-3 part that carries the macroscopic quantities and $\boldsymbol{A}_2$ is the remainder. To make subsequent computations efficient, it is desirable for the correction $\boldsymbol{X}$ to be low-rank as well. A standard approach is to use nuclear norm minimization [3] as a convex relaxation of rank minimization, which leads to the formulation

$$\begin{aligned} &\min_{\boldsymbol{X}} \|\boldsymbol{X}\|_* + \frac{a}{2}\|\boldsymbol{X} - \boldsymbol{A}_2\|_F^2, \\ &\text{subject to:} \quad \boldsymbol{X} \geqslant -\boldsymbol{A}_1,\ \boldsymbol{X}\boldsymbol{B} = \boldsymbol{0}, \end{aligned} \tag{1}$$

where $\|\cdot\|_*$ denotes the nuclear norm (sum of singular values) and $a > 0$ is a penalty parameter that controls the trade-off between low rank and approximation error, and the corrected solution is $\boldsymbol{A}_1 + \boldsymbol{X}$. However, this formulation has two drawbacks. It is not consistent, since if $\boldsymbol{A}$ is already nonnegative, the solution is not necessarily the identity correction. It is also not homogeneous, since scaling the input does not scale the output. To address these two issues, we reinterpret $\boldsymbol{X}$ as a correction term added to $\boldsymbol{A}$ rather than an approximation to $\boldsymbol{A}_2$, so the Frobenius term penalizes the magnitude of the correction itself rather than an approximation error. We also replace the nuclear norm by its square, which makes the objective homogeneous of degree two. This leads to the formulation

$$\begin{aligned} &\min_{\boldsymbol{X}} \|\boldsymbol{X}\|_*^2 + \frac{a}{2}\|\boldsymbol{X}\|_F^2, \\ &\text{subject to:} \quad \boldsymbol{X} \geqslant -\boldsymbol{A},\ \boldsymbol{X}\boldsymbol{B} = \boldsymbol{0}, \end{aligned} \tag{2}$$

where $\boldsymbol{X}$ is now a correction term and the corrected solution is $\boldsymbol{A} + \boldsymbol{X}$. This formulation is both consistent and homogeneous. One major computational challenge is to evaluate the proximal operator of the squared nuclear norm under the orthogonality constraint $\boldsymbol{X}\boldsymbol{B} = \boldsymbol{0}$. For the standard nuclear norm in (1), the proximal operator is the singular value soft thresholding operator [3]. For the squared nuclear norm in (2), the proximal operator can still be expressed as singular value soft thresholding, but the threshold is now implicit: it depends on the nuclear norm of the unknown minimizer itself. We derive this characterization in Theorem 1 and show that the implicit threshold can be computed by a bisection search that requires only $\mathcal{O}(\log(\min(m, n)))$ singular value comparisons for an $m \times n$ matrix.

We consider two formulations for the nonnegative correction problem. For the convex problem (2), we develop five algorithms: Douglas–Rachford splitting [17] on the primal problem, restarted dual FISTA [2], restarted dual accelerated gradient descent, dual Polak–Ribière plus conjugate gradient, and dual L-BFGS [19]. These algorithms converge to a global minimizer. We also consider

a non-convex formulation with an explicit rank constraint on the correction term. For this formulation, we develop a tangent-space accelerated alternating projection (TAP) algorithm inspired by [26]. The key idea is to work on the tangent space of the low-rank manifold so that each iteration only requires a $2r \times 2r$ SVD for a rank r solution instead of a full SVD. Beyond the fixed correction problem, we demonstrate the correction as a positivity limiter inside the time-dependent LoMaC conservative low-rank scheme of [14]. Applied periodically as the solution is advanced, it removes the negativity introduced by the SVD-type truncation while leaving the conserved moments untouched, so that mass, momentum, and energy remain conserved.

The rest of the paper is organized as follows. In Section 2, we describe the problem background, including the low-rank solution structure for Vlasov dynamics and the conservative orthogonal projection from [13, 14]. In Section 3, we formulate the nonnegative low-rank matrix correction problem with orthogonality constraint and present three optimization formulations. In Section 4, we develop the numerical algorithms for both the convex and non-convex formulations. In Section 5, we present numerical results on a Landau damping test case that demonstrate the effectiveness and compare the performance of all proposed algorithms, study their empirical cost scaling with problem size, and demonstrate the correction as a positivity limiter in a time-dependent conservative low-rank Vlasov solver. We conclude in Section 6.

2 Low-rank Vlasov dynamics and the orthogonality constraint

2.1 Low-rank Vlasov solutions

We consider a 1D1V Vlasov system [12]

$$\begin{aligned}
&\frac{\partial f}{\partial t} + v \cdot \nabla_x f - E(x,t) \cdot \nabla_v f = 0, \quad \Omega_x \times \Omega_v \times (0,T], \\
&-\Delta_x \phi = \rho(x,t), \quad \rho(x,t) = \rho_0 - \int_{\mathbb{R}^{d_v}} f(x,v,t)\,\mathrm{d}v, \quad \Omega_x \times (0,T], \\
&E(x,t) = -\nabla_x \phi, \quad \Omega_x \times (0,T],
\end{aligned}$$

where f is the probability density function of charged particles, with a corresponding charge density ρ, which generates a self-consistent electrostatic potential ϕ via the Poisson equation. The Vlasov equation is discretized on the phase space $[x_{\min}, x_{\max}] \times [-v_{\max}, v_{\max}]$ with an equidistant $N_x \times N_v$ Cartesian grid

$$\begin{aligned}
x_{\text{grid}} &: x_{\min} = x_1 < \cdots < x_{N_x} = x_{\max}, \\
v_{\text{grid}} &: -v_{\max} = v_1 < \cdots < v_{N_v} = v_{\max}.
\end{aligned}$$

We denote the mesh size in the v-direction by h_v. The numerical solution $\boldsymbol{F} \in \mathbb{R}^{N_x \times N_v}$, with $\boldsymbol{F}_{i,j} \approx f(x_i, v_j)$ approximating the probability distribution function $f(x,v)$, is stored in a low-rank format

$$\boldsymbol{F} = \boldsymbol{F}_1 + \boldsymbol{F}_2 = \boldsymbol{U}_1 \boldsymbol{V}_1^T + \boldsymbol{U}_2 \boldsymbol{V}_2^T, \tag{3}$$

where $\boldsymbol{F}_1$ is a rank-3 term that carries the three macroscopic quantities (density, momentum, and internal energy) of $\boldsymbol{F}$, and $\boldsymbol{F}_2$ is the remainder. This decomposition is the key structure underlying the conservative low-rank tensor methods in [13, 14].

The three macroscopic quantities are defined as velocity moments of $f(x, v)$:

$$\rho(x) = \int f(x, v)\mathrm{d}v, \tag{4a}$$
$$m(x) = \int v f(x, v)\mathrm{d}v, \tag{4b}$$
$$e(x) = \int v^2 f(x, v)\mathrm{d}v. \tag{4c}$$

Let $\mathbf{1} = (1, \ldots, 1) \in \mathbb{R}^{1\times N_v}$, $\boldsymbol{v} = (v_1, \ldots, v_{N_v}) \in \mathbb{R}^{1\times N_v}$, and $\boldsymbol{v}^{\circ 2} = (v_1^2, \ldots, v_{N_v}^2) \in \mathbb{R}^{1\times N_v}$. The discrete macroscopic quantities are computed as

$$\boldsymbol{\rho} = h_v \boldsymbol{F}\mathbf{1}^T = h_v \boldsymbol{F}_1 \mathbf{1}^T, \tag{5a}$$
$$\boldsymbol{m} = h_v \boldsymbol{F}\boldsymbol{v}^T = h_v \boldsymbol{F}_1 \boldsymbol{v}^T, \tag{5b}$$
$$\boldsymbol{e} = h_v \boldsymbol{F}(\boldsymbol{v}^{\circ 2})^T = h_v \boldsymbol{F}_1 (\boldsymbol{v}^{\circ 2})^T, \tag{5c}$$

where the second equalities hold by the requirement that $\boldsymbol{F}_1$ carries all three macroscopic quantities, i.e., $\boldsymbol{F}_2$ has zero velocity moments.

2.2 Conservative orthogonal decomposition

Following [13], we construct the decomposition $\boldsymbol{F} = \boldsymbol{F}_1 + \boldsymbol{F}_2$ using a weighted inner product in the velocity space. Let

$$\boldsymbol{R} = \begin{pmatrix} \mathbf{1} \\ \boldsymbol{v} \\ \boldsymbol{v}^{\circ 2} \end{pmatrix} \in \mathbb{R}^{3\times N_v} \tag{6}$$

be the matrix whose rows span the space of velocity moments, and let

$$\boldsymbol{S} = \mathrm{diag}\,(w(v_1), w(v_2), \cdots, w(v_{N_v})) \in \mathbb{R}^{N_v\times N_v} \tag{7}$$

be a diagonal weight matrix with pointwise-positive weight function $w(v) > 0$. The requirement that $\boldsymbol{F}_2$ has zero velocity moments is expressed as

$$\boldsymbol{F}_2\boldsymbol{R}^T = \mathbf{0}, \qquad \text{or equivalently,} \qquad \boldsymbol{F}_1\boldsymbol{R}^T = \boldsymbol{F}\boldsymbol{R}^T. \tag{8}$$

Additionally, we require $\mathrm{rank}(\boldsymbol{F}_1) = 3$ and that the row space of $\boldsymbol{F}_1$ lies in the column space of $\boldsymbol{RS}$. This gives the system

$$\begin{cases} \boldsymbol{F}_1 = \boldsymbol{LRS}, \\ \boldsymbol{F}_1\boldsymbol{R}^T = \boldsymbol{F}\boldsymbol{R}^T, \end{cases} \tag{9}$$

where $\boldsymbol{L} \in \mathbb{R}^{N_x\times 3}$. To solve for $\boldsymbol{F}_1$ explicitly, we compute a QR decomposition

$$\sqrt{\boldsymbol{S}}\,\boldsymbol{R}^T = \boldsymbol{B}\widetilde{\boldsymbol{R}}, \tag{10}$$

where $\boldsymbol{B} \in \mathbb{R}^{N_v\times 3}$ has orthonormal columns and $\widetilde{\boldsymbol{R}} \in \mathrm{GL}_3(\mathbb{R})$. Then $\boldsymbol{F}_1$ is given by

$$\boldsymbol{F}_1 = \boldsymbol{F}\sqrt{\boldsymbol{S}}^{-1}\boldsymbol{B}\boldsymbol{B}^T\sqrt{\boldsymbol{S}}. \tag{11}$$

Thus, $\boldsymbol{F}_1$ is the orthogonal projection of $\boldsymbol{F}$ (with respect to the weighted inner product defined by $\boldsymbol{S}$) onto the three-dimensional subspace spanned by the velocity moment basis functions $\mathbf{1}, \boldsymbol{v}, \boldsymbol{v}^{\circ 2}$.

2.3 Rescaled problem and the orthogonality constraint

To formulate the correction problem, we introduce the rescaled matrices

$$\boldsymbol{A}_1 = \boldsymbol{F}_1 \sqrt{\boldsymbol{S}}^{-1}, \tag{12a}$$

$$\boldsymbol{A}_2 = \boldsymbol{F}_2 \sqrt{\boldsymbol{S}}^{-1}, \tag{12b}$$

$$\boldsymbol{A} = \boldsymbol{A}_1 + \boldsymbol{A}_2 = \boldsymbol{F} \sqrt{\boldsymbol{S}}^{-1}. \tag{12c}$$

The rescaling converts the weighted inner product to the standard Frobenius inner product, so that the orthogonal projection (11) takes the simple form $\boldsymbol{A}_1 = \boldsymbol{A}\boldsymbol{B}\boldsymbol{B}^T$. Note that $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$ inherit the low-rank structure of $\boldsymbol{F}_1$ and $\boldsymbol{F}_2$, and the nonnegativity of $\boldsymbol{F}$ is equivalent to the nonnegativity of $\boldsymbol{A}$ (since $\boldsymbol{S}$ has positive diagonal entries).

A remaining issue for adaptive rank approaches is that the matrix $\boldsymbol{A}$ (or equivalently $\boldsymbol{F}$) may contain negative entries. We seek a correction term $\boldsymbol{X}$ such that $\boldsymbol{A} + \boldsymbol{X} \geqslant 0$. We also require the corrected solution $\boldsymbol{A}+\boldsymbol{X}$ to have the same macroscopic quantities as $\boldsymbol{A}$. Since $\boldsymbol{A}_1 = \boldsymbol{A}\boldsymbol{B}\boldsymbol{B}^T$ carries the macroscopic information, the requirement that the correction $\boldsymbol{X}$ does not alter the macroscopic quantities is equivalent to

$$\boldsymbol{X}\boldsymbol{B} = \boldsymbol{0}. \tag{13}$$

This is the orthogonality constraint: the correction term $\boldsymbol{X}$ must lie in the orthogonal complement of the column space of $\boldsymbol{B}$. In the following, we use $\boldsymbol{B} \in \mathbb{R}^{N_v \times 3}$ to denote the matrix with orthonormal columns defined in this section.

3 Nonnegative low-rank matrix correction: optimization formulations

With the setup from the previous section, the goal is to find a correction term $\boldsymbol{X}$ such that the corrected solution $\boldsymbol{A}+\boldsymbol{X}$ is nonnegative and preserves the macroscopic quantities. The constraints are

$$\boldsymbol{A} + \boldsymbol{X} \geqslant 0, \qquad \boldsymbol{X}\boldsymbol{B} = \boldsymbol{0}.$$

Ideally, the correction $\boldsymbol{X}$ should have low rank so that the corrected solution retains a compact low-rank representation. We present three optimization formulations for this problem.

3.1 The approximation formulation

We first seek a nonnegative low-rank approximation $\boldsymbol{X}$ to the remainder $\boldsymbol{A}_2$, so that the corrected solution $\boldsymbol{A}_1 + \boldsymbol{X}$ is nonnegative and preserves the macroscopic quantities. The ideal problem is

$$\min_{\boldsymbol{X}} \operatorname{rank}(\boldsymbol{X}) \quad \text{and} \quad \|\boldsymbol{X} - \boldsymbol{A}_2\|_F, \quad \text{subject to} \quad \boldsymbol{X} + \boldsymbol{A}_1 \geqslant 0,\ \boldsymbol{X}\boldsymbol{B} = \boldsymbol{0}. \tag{14}$$

However, rank minimization is NP-hard in general. A standard convex relaxation technique is to replace the rank function by the nuclear norm [3].

Definition 1 (nuclear norm)**.** *For any matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$, its nuclear norm is defined as*

$$\|\boldsymbol{A}\|_* = \sum_{k=1}^{\min\{m,n\}} \sigma_k, \tag{15}$$

where σ_k is the k-th singular value of $\boldsymbol{A}$.

The nuclear norm is the tightest convex lower bound of the rank function on the unit spectral norm ball. Minimizing the nuclear norm therefore promotes low-rank solutions in the same way that minimizing the ℓ^1 norm of a vector promotes sparsity. We record a standard property that will be used in the algorithm derivation, using $\|\cdot\|_2$ to denote the spectral norm.

Proposition 1. *Suppose the compact SVD of $\boldsymbol{X} \in \mathbb{R}^{m\times n}$ is $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$. Then the subdifferential of $\|\cdot\|_*$ at $\boldsymbol{X}$ is*

$$\partial \|\boldsymbol{X}\|_* = \left\{\boldsymbol{U}\boldsymbol{V}^T + \boldsymbol{W} \;\middle|\; \boldsymbol{U}^T\boldsymbol{W} = \boldsymbol{0},\ \boldsymbol{W}\boldsymbol{V} = \boldsymbol{0},\ \|\boldsymbol{W}\|_2 \leqslant 1\right\}. \tag{16}$$

Using the nuclear norm as a convex surrogate for the rank, and choosing the Frobenius norm for the approximation error, we obtain the following convex optimization problem:

$$\begin{aligned} &\min_{\boldsymbol{X}} \|\boldsymbol{X}\|_* + \frac{a}{2}\|\boldsymbol{X} - \boldsymbol{A}_2\|_F^2, \\ &\text{subject to:} \quad \boldsymbol{X} \geqslant -\boldsymbol{A}_1,\ \boldsymbol{X}\boldsymbol{B} = \boldsymbol{0}, \end{aligned}$$

where $a > 0$ is a parameter that controls the trade-off between low rank and approximation accuracy. This is precisely the formulation in (1).

3.2 The correction formulation (convex)

The formulation (1) has two drawbacks. First, it is not consistent: if $\boldsymbol{A}$ is already nonnegative, the minimizer $\boldsymbol{X}$ of (1) is not necessarily $\boldsymbol{A}_2$, meaning that the post-processing may alter a solution that needs no correction. This can be seen from the following example.

Example 1. *Drop the orthogonality constraint and consider the diagonal case: $\boldsymbol{A}_1 = \boldsymbol{0}$ and $\boldsymbol{A}_2 = \operatorname{diag}(a_1, \cdots, a_n)$ with $a_i \geqslant 0$. Then $\|\boldsymbol{X}\|_* = \sum_i |x_i|$ reduces to the ℓ^1 norm of the diagonal, and singular value thresholding acts entrywise, so the minimizer of (1) is $\boldsymbol{X} = \operatorname{diag}(x_1, \cdots, x_n)$ with*

$$x_i = \max\left\{a_i - \frac{1}{a},\ 0\right\},$$

which differs from $\boldsymbol{A}_2$ unless $a_i \geqslant 1/a$ for all i.

Second, the formulation is not homogeneous: if the input is scaled by $\alpha > 0$, the minimizer does not scale by the same factor. This can also be verified through Example 1.

To address these two issues, we seek a *correction term* $\boldsymbol{X}$ to be added to $\boldsymbol{A}$, rather than an approximation to $\boldsymbol{A}_2$. We also replace the nuclear norm by its square, which makes the objective homogeneous of degree two. This leads to the formulation

$$\begin{aligned} &\min_{\boldsymbol{X}} \|\boldsymbol{X}\|_*^2 + \frac{a}{2}\|\boldsymbol{X}\|_F^2, \\ &\text{subject to:} \quad \boldsymbol{X} \geqslant -\boldsymbol{A},\ \boldsymbol{X}\boldsymbol{B} = \boldsymbol{0}. \end{aligned}$$

which is the formulation (2). Here $\boldsymbol{X}$ is the correction term and the corrected solution is $\boldsymbol{A}+\boldsymbol{X}$. The constraint $\boldsymbol{X} \geqslant -\boldsymbol{A}$ ensures nonnegativity of the corrected solution, and the constraint $\boldsymbol{X}\boldsymbol{B} = \boldsymbol{0}$ ensures preservation of the macroscopic quantities. The squared nuclear norm $\|\boldsymbol{X}\|_*^2$ promotes low

rank (since $\operatorname{rank}(\boldsymbol{X}+\boldsymbol{Y}) \leqslant \operatorname{rank}(\boldsymbol{X}) + \operatorname{rank}(\boldsymbol{Y})$, a small correction should have low rank), while the Frobenius norm penalty $\frac{a}{2}\|\boldsymbol{X}\|_F^2$ penalizes large corrections. We refer to (2) as the convex formulation in the rest of the paper.

This formulation is consistent: if $\boldsymbol{A} \geqslant 0$, then $\boldsymbol{X} = \boldsymbol{0}$ is clearly the minimizer. It is also homogeneous: if the input is $\alpha\boldsymbol{A}$, the minimizer is $\alpha\boldsymbol{X}$.

Remark 1. *In practice, we have not observed significant differences in the numerical results between the formulations* (1) *and* (2) *when the parameter a is well chosen.*

Remark 2 (Non-emptiness of the admissible set)**.** *The admissible set* $\left\{\boldsymbol{X} \in \mathbb{R}^{m\times n} \middle| \boldsymbol{X} \geqslant -\boldsymbol{A},\ \boldsymbol{X}\boldsymbol{B} = \boldsymbol{0}\right\}$ *may be empty. It is nonempty if and only if there exists a nonnegative matrix $\widetilde{\boldsymbol{A}} \in \mathbb{R}^{m\times n}$ with $\widetilde{\boldsymbol{A}}\boldsymbol{B}\boldsymbol{B}^T = \boldsymbol{A}\boldsymbol{B}\boldsymbol{B}^T$, i.e., the macroscopic part of $\boldsymbol{A}$ coincides with the macroscopic part of some nonnegative distribution. In the Vlasov dynamics setting, this condition is satisfied whenever the macroscopic moments (ρ, m, e) are realizable, i.e., there exists a nonnegative distribution with these moments. For the first three moments, a necessary and sufficient condition is $\rho > 0$ and $e\rho \geqslant m^2$.*

3.3 The rank-constrained formulation (non-convex)

In some applications, it is desirable to explicitly control the rank of the correction term so that the corrected solution can be stored efficiently in a low-rank format. For a given target rank r, we consider the non-convex formulation

$$\begin{aligned} &\min_{\boldsymbol{X}} \|\boldsymbol{X}\|_F^2, \\ &\text{subject to:} \quad \operatorname{rank}(\boldsymbol{X}) \leqslant r,\ \boldsymbol{X} \geqslant -\boldsymbol{A},\ \boldsymbol{X}\boldsymbol{B} = \boldsymbol{0}. \end{aligned} \tag{17}$$

Unlike the convex formulations (1) and (2), the rank constraint makes the feasible set non-convex, so global optimality cannot be guaranteed. However, this formulation has the computational advantage that the iterates can be maintained in a rank-r factored form throughout the algorithm, as we will see in Section 4. We refer to (17) as the non-convex formulation in the rest of the paper.

4 Algorithms

In this section, we develop algorithms for both the convex formulation (2) and the non-convex formulation (17). For the convex problem, we present five algorithms in Sections 4.2–4.6: Douglas–Rachford splitting on the primal problem, restarted dual FISTA, restarted dual accelerated gradient descent, dual PR+ conjugate gradient, and dual L-BFGS. For the non-convex problem, we develop a tangent-space accelerated alternating projection algorithm in Section 4.7. We begin with a brief review of the optimization tools used throughout this section. Readers familiar with convex optimization may skip to Section 4.2.

4.1 Convex optimization preliminaries

The convex formulation (2) involves minimizing a non-smooth objective (the squared nuclear norm plus a Frobenius penalty) subject to non-smooth constraints (nonnegativity and orthogonality). We briefly review the key concepts needed for the algorithms.

Indicator functions and subdifferentials. To incorporate constraints into an unconstrained framework, we use the indicator function of a convex set $\mathcal{C}$:

$$I_{\mathcal{C}}(\boldsymbol{X}) = \begin{cases} 0 & \text{if } \boldsymbol{X} \in \mathcal{C}, \\ +\infty & \text{otherwise,} \end{cases}$$

so that $\min_{\boldsymbol{X}\in\mathcal{C}} \varphi(\boldsymbol{X}) = \min_{\boldsymbol{X}} \varphi(\boldsymbol{X}) + I_{\mathcal{C}}(\boldsymbol{X})$. For a convex but non-differentiable function φ, the gradient is replaced by the subdifferential $\partial\varphi(\boldsymbol{X}_0)$, defined as the set of all $\boldsymbol{G}$ satisfying $\varphi(\boldsymbol{Y}) \geqslant \varphi(\boldsymbol{X}_0) + \langle \boldsymbol{G}, \boldsymbol{Y} - \boldsymbol{X}_0 \rangle$ for all $\boldsymbol{Y}$, where the inner product is defined $\langle \boldsymbol{Y}, \boldsymbol{X} \rangle = \operatorname{tr}(\boldsymbol{Y}^T\boldsymbol{X})$. When φ is differentiable, $\partial\varphi = \{\nabla\varphi\}$. As a scalar example, $\partial|x|(0) = [-1, 1]$. The subdifferential of the nuclear norm is given in Proposition 1.

Proximal operators. Many convex optimization problems can be interpreted as finding the steady state of the (sub)gradient flow

$$\dot{\boldsymbol{X}} \in -\partial\varphi(\boldsymbol{X}),$$

where $\partial\varphi$ denotes the subdifferential of a proper, closed, convex function φ. When φ is differentiable, this reduces to the familiar gradient flow

$$\dot{\boldsymbol{X}} = -\nabla\varphi(\boldsymbol{X}).$$

Applying the backward Euler method with time step Δt gives

$$\frac{\boldsymbol{X}^{k+1} - \boldsymbol{X}^k}{\Delta t} \in -\partial\varphi(\boldsymbol{X}^{k+1}),$$

or equivalently,

$$\boldsymbol{0} \in \partial\varphi(\boldsymbol{X}^{k+1}) + \frac{1}{\Delta t}(\boldsymbol{X}^{k+1} - \boldsymbol{X}^k).$$

Remarkably, this is precisely the first-order optimality condition of the optimization problem

$$\boldsymbol{X}^{k+1} = \arg\min_{\boldsymbol{X}} \left\{ \varphi(\boldsymbol{X}) + \frac{1}{2\Delta t}\|\boldsymbol{X} - \boldsymbol{X}^k\|_F^2 \right\}. \tag{18}$$

Identifying $\eta = 1/\Delta t$ yields the proximal operator

$$\operatorname{prox}_{\eta^{-1}\varphi}(\boldsymbol{Z}) = \arg\min_{\boldsymbol{X}} \left\{ \varphi(\boldsymbol{X}) + \frac{\eta}{2}\|\boldsymbol{X} - \boldsymbol{Z}\|_F^2 \right\},$$

which may therefore be viewed as a single implicit (backward Euler) step of the subgradient flow. Consequently, the proximal point iteration

$$\boldsymbol{X}^{k+1} = \operatorname{prox}_{\eta^{-1}\varphi}(\boldsymbol{X}^k)$$

inherits the unconditional stability characteristic of backward Euler and converges to a minimizer of φ under standard assumptions. An important special case is the indicator function of a closed convex set,

$$I_{\mathcal{C}}(\boldsymbol{X}) = \begin{cases} 0, & \boldsymbol{X} \in \mathcal{C}, \\ +\infty, & \text{otherwise,} \end{cases}$$

whose proximal operator reduces to the Euclidean projection,

$$\operatorname{prox}_{\eta^{-1}I_{\mathcal{C}}} = \Pi_{\mathcal{C}},$$

independent of η.

Splitting method. Many optimization problems take the composite form

$$\min_{\boldsymbol{X}} \; f(\boldsymbol{X}) + g(\boldsymbol{X}),$$

where different terms represent different objectives or constraints. The choice of numerical method depends on the structure of these two components. When both f and g are differentiable, their gradients can be combined,

$$\nabla(f+g) = \nabla f + \nabla g,$$

and standard gradient-based methods apply. If only one term (say f) is differentiable while g is nonsmooth, then one may take an explicit gradient step on f followed by an implicit (proximal) step on g, leading to the proximal-gradient method. The more challenging situation arises when both f and g are nonsmooth. In principle, one could apply the proximal point method directly to the sum,

$$\mathrm{prox}_{\eta^{-1}(f+g)},$$

but this generally requires solving a difficult optimization problem involving both functions simultaneously. In contrast, the individual proximal operators,

$$\mathrm{prox}_{\eta^{-1}f} \quad \text{and} \quad \mathrm{prox}_{\eta^{-1}g},$$

are often available in closed form or can be computed efficiently because each function possesses a much simpler structure. Proximal splitting methods exploit this observation by replacing one difficult proximal evaluation of $f+g$ with a sequence of simpler proximal evaluations involving only f and g. This philosophy is analogous to operator splitting methods for differential equations, where a complicated evolution operator is decomposed into simpler subproblems that can be solved individually. Douglas–Rachford splitting is one of the most successful examples of such proximal splitting methods, alternating between the proximal operators of f and g while still converging to a minimizer of the composite problem. Our convex problem (2) falls precisely into this setting. The nonnegativity constraint and the orthogonality-constrained nuclear norm are both nonsmooth, making the proximal operator of their sum difficult to compute directly. However, each individual proximal operator admits an efficient closed-form solution, making Douglas–Rachford splitting a natural and effective approach.

4.2 Douglas–Rachford splitting for the convex problem

4.2.1 Splitting

To apply Douglas–Rachford splitting, the objective should be decomposed into two convex functions whose proximal operators are individually inexpensive to evaluate. The goal is therefore not to split the objective arbitrarily, but rather to isolate the different structures so that each proximal subproblem admits either a closed-form solution or an efficient numerical algorithm.

For our problem, the two nonsmooth components arise from the nonnegativity constraint and the orthogonality-constrained nuclear norm. These two structures are naturally handled by different proximal operators, leading to the decomposition

$$h(\boldsymbol{X}) = f(\boldsymbol{X}) + g(\boldsymbol{X}), \tag{19a}$$

$$f(\boldsymbol{X}) = \tfrac{a_1}{2}\,\|\boldsymbol{X}\|_F^2 + I_{\{\boldsymbol{X}\,|\,\boldsymbol{X}\geqslant -\boldsymbol{A}\}}(\boldsymbol{X}), \tag{19b}$$

$$g(\boldsymbol{X}) = \|\boldsymbol{X}\|_*^2 + \tfrac{a_2}{2}\|\boldsymbol{X}\|_F^2 + I_{\{\boldsymbol{X}\mid \boldsymbol{X}\boldsymbol{B}=\boldsymbol{0}\}}(\boldsymbol{X}), \tag{19c}$$

where $a_1 + a_2 = a$.

The quadratic regularization is split between f and g through the parameters a_1 and a_2. This decomposition preserves the original objective while allowing the proximal operators of both subproblems to remain well conditioned. In particular,

- f consists of a quadratic term together with the box constraint $\boldsymbol{X} \geq -\boldsymbol{A}$. Its proximal operator is completely separable over the matrix entries and reduces to an elementwise projection.
- g combines the squared nuclear norm and the linear orthogonality constraint $\boldsymbol{X}\boldsymbol{B} = \boldsymbol{0}$. Although this proximal operator is more involved, the orthogonality constraint has a simple linear structure, allowing a closed-form solution to be derived through the optimality conditions.

Since both f and g are proper, closed, and convex, Douglas–Rachford splitting is applicable. Instead of solving one difficult proximal problem involving

$$\operatorname{prox}_{\eta^{-1}(f+g)},$$

the algorithm alternates between the much simpler proximal operators

$$\operatorname{prox}_{\eta^{-1}f} \quad \text{and} \quad \operatorname{prox}_{\eta^{-1}g},$$

while still converging to a minimizer of the original convex problem.

Algorithm 1: DR for convex problem

Input: $\boldsymbol{Z}_1$, $\boldsymbol{A}$, $\boldsymbol{B}$, $a_1, a_2 \geqslant 0$, $\eta > 0$, $\alpha \in (0, 1]$, maxit
Output: $\boldsymbol{X}_{\text{maxit}}$

```
for k = 1 : maxit do
    X_k ← arg min_X f(X) + (η/2) ‖X − Z_k‖_F^2 ;        // first proximal operator
    Y_k ← arg min_Y g(Y) + (η/2) ‖Y − 2X_k + Z_k‖_F^2 ;  // second proximal operator
    Z_{k+1} ← Z_k + 2α (Y_k − X_k);
end
```

The proximal operator of f has a simple closed form. Since f is a quadratic plus an indicator function for the set $\{\boldsymbol{X} \mid \boldsymbol{X} \geqslant -\boldsymbol{A}\}$, the problem is separable and the proximal operator is an elementwise clipping:

$$\arg\min_{\boldsymbol{X}} f(\boldsymbol{X}) + \frac{\eta}{2}\|\boldsymbol{X} - \boldsymbol{Z}_k\|_F^2 = \pi_+\left(\frac{\eta}{a_1 + \eta}\boldsymbol{Z}_k\right), \tag{20}$$

where

$$\pi_+(\boldsymbol{Z})_{ij} = \max\{-A_{ij}, Z_{ij}\}. \tag{21}$$

4.2.2 The proximal operator of the orthogonality-constrained squared nuclear norm

Unlike the proximal operator of f, which is completely separable over the matrix entries, the proximal operator of g is substantially more challenging. The difficulty arises from two sources. First, the squared nuclear norm

$$\|\boldsymbol{Y}\|_*^2 = \left(\sum_i \sigma_i(\boldsymbol{Y})\right)^2$$

couples all singular values, so the optimization problem is no longer separable. Second, the orthogonality constraint

$$\boldsymbol{Y}\boldsymbol{B} = \boldsymbol{0}$$

introduces a linear constraint on the admissible right singular vectors. Consequently, unlike the proximal operator of f, no elementwise solution is available. Our strategy is therefore different. Rather than solving the minimization problem directly, we first derive its first-order optimality condition, identify its structure, and then show that this condition is equivalent to a modified singular-value thresholding problem. This ultimately leads to a closed-form characterization of the proximal operator.

The optimality condition for

$$\boldsymbol{Y} = \arg\min_{\boldsymbol{Y}} g(\boldsymbol{Y}) + \frac{\eta}{2}\|\boldsymbol{Y} - \boldsymbol{M}\|_F^2 \tag{22}$$

is

$$\begin{cases} \boldsymbol{0} \in \partial\|\boldsymbol{Y}\|_*^2 + (a_2+\eta)\boldsymbol{Y} - \eta\boldsymbol{M} + \left\{\boldsymbol{\Lambda}\boldsymbol{B}^T \mid \boldsymbol{\Lambda} \in \mathbb{R}^{N_x\times 3}\right\}, \\ \boldsymbol{Y}\boldsymbol{B} = \boldsymbol{0}. \end{cases} \tag{23}$$

We first state a lemma for the subgradient of the squared nuclear norm.

Lemma 1. $2\|\boldsymbol{X}\|_* \partial\|\boldsymbol{X}\|_* \subset \partial\|\boldsymbol{X}\|_*^2$.

Proof. For each $\boldsymbol{Z} \in \partial\|\boldsymbol{X}\|_*$, we have $\|\boldsymbol{Y}\|_* - \|\boldsymbol{X}\|_* \geqslant \langle \boldsymbol{Z}, \boldsymbol{Y} - \boldsymbol{X}\rangle$ for all $\boldsymbol{Y}$. Then

$$\begin{aligned} \|\boldsymbol{Y}\|_*^2 - \|\boldsymbol{X}\|_*^2 - 2\|\boldsymbol{X}\|_*\langle \boldsymbol{Z}, \boldsymbol{Y} - \boldsymbol{X}\rangle &\geqslant \|\boldsymbol{Y}\|_*^2 - \|\boldsymbol{X}\|_*^2 - 2\|\boldsymbol{X}\|_*(\|\boldsymbol{Y}\|_* - \|\boldsymbol{X}\|_*) \\ &= (\|\boldsymbol{Y}\|_* - \|\boldsymbol{X}\|_*)^2 \geqslant 0. \end{aligned}$$

Therefore, $2\|\boldsymbol{X}\|_* \boldsymbol{Z} \in \partial\|\boldsymbol{X}\|_*^2$. □

Definition 2. *For any matrix $\boldsymbol{O} \in \mathbb{R}^{n\times k}$ $(k \leqslant n)$ with orthonormal columns, we define the orthogonal projection and its complement as*

$$\boldsymbol{\Pi}_{\boldsymbol{O}} = \boldsymbol{O}\boldsymbol{O}^T, \qquad \boldsymbol{\Pi}_{\boldsymbol{O}}^{\perp} = \boldsymbol{I} - \boldsymbol{O}\boldsymbol{O}^T. \tag{24}$$

The next theorem characterizes the proximal operator of g. It shows that the proximal operator can be expressed in terms of the singular value soft thresholding operator $D_\tau(\cdot)$, which subtracts τ from each singular value and discards the non-positive ones.

Theorem 1. *Suppose the compact SVD of the minimizer $\boldsymbol{Y}$ of (22) is $\boldsymbol{Y} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$. Then $\boldsymbol{Y}$ satisfies*

$$\boldsymbol{Y} = D_{\frac{2\|\boldsymbol{Y}\|_*}{a_2+\eta}}\left(\frac{\eta}{a_2+\eta}\boldsymbol{M}\boldsymbol{\Pi}_{\boldsymbol{B}}^{\perp}\right), \tag{25}$$

where $D_\tau(\cdot)$ is the singular value soft thresholding operator with threshold τ.

Proof. Since g is strongly convex, problem (22) has a unique minimizer. We construct a candidate $\boldsymbol{Y}$ via (25) and verify that it satisfies the first-order optimality condition (23). By Lemma 1 and Proposition 1, $\boldsymbol{Y}$ satisfies (23) if the following system admits a solution:

$$\begin{aligned}&\exists \boldsymbol{Z} \in \mathbb{R}^{N_x \times 3}, \boldsymbol{W} \in \left\{\boldsymbol{W} \in \mathbb{R}^{N_x \times N_v} | \boldsymbol{U}^T \boldsymbol{W} = \boldsymbol{0}, \boldsymbol{W}\boldsymbol{V} = \boldsymbol{0}, \|\boldsymbol{W}\|_2 \leqslant 1\right\}: \\ &\begin{cases} \boldsymbol{0} = 2\left\|\boldsymbol{\Sigma}\right\|_* \boldsymbol{U}\boldsymbol{V}^T + 2\left\|\boldsymbol{\Sigma}\right\|_* \boldsymbol{W} + \boldsymbol{Z}\boldsymbol{B}^T + (a_2+\eta)\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T - \eta \boldsymbol{M}, \\ \boldsymbol{V}^T\boldsymbol{B} = \boldsymbol{0}, \end{cases}\end{aligned} \tag{26}$$

which can be rewritten as

$$\begin{aligned}&\exists \boldsymbol{Z} \in \mathbb{R}^{N_x \times 3}, \boldsymbol{W} \in \left\{\boldsymbol{W} \in \mathbb{R}^{N_x \times N_v} | \boldsymbol{U}^T \boldsymbol{W} = \boldsymbol{0}, \boldsymbol{W}\boldsymbol{V} = \boldsymbol{0}, \|\boldsymbol{W}\|_2 \leqslant 1\right\}: \\ &\begin{cases} \frac{\eta}{a_2+\eta}\boldsymbol{M} = \boldsymbol{U}\left(\boldsymbol{\Sigma} + \frac{2\|\boldsymbol{\Sigma}\|_*}{a_2+\eta}\right)\boldsymbol{V}^T + \frac{2\|\boldsymbol{\Sigma}\|_*}{a_2+\eta}\boldsymbol{W} + \boldsymbol{Z}\boldsymbol{B}^T, \\ \boldsymbol{V}^T\boldsymbol{B} = \boldsymbol{0}. \end{cases}\end{aligned} \tag{27}$$

We now show that this system admits a solution. This proves that the candidate $\boldsymbol{Y}$ constructed via (25) satisfies the optimality condition and is therefore the unique minimizer.

Let $\tau = \frac{2\|\boldsymbol{\Sigma}\|_*}{a_2+\eta}$, then

$$\begin{aligned}\frac{\eta}{a_2+\eta}\boldsymbol{M} &= \boldsymbol{U}\left(\boldsymbol{\Sigma}+\tau\right)\boldsymbol{V}^T + \tau\boldsymbol{W} + \boldsymbol{Z}\boldsymbol{B}^T \\ &= \boldsymbol{U}\left(\boldsymbol{\Sigma}+\tau\right)\boldsymbol{V}^T + \tau\boldsymbol{W}(\boldsymbol{I}-\boldsymbol{B}\boldsymbol{B}^T) + \left(\boldsymbol{Z}+\tau\boldsymbol{W}\boldsymbol{B}\right)\boldsymbol{B}^T \\ &= \begin{pmatrix}\boldsymbol{U} & \widetilde{\boldsymbol{U}}\end{pmatrix}\begin{pmatrix}\boldsymbol{\Sigma}+\tau & \boldsymbol{0} \\ \boldsymbol{0} & \tau\widetilde{\boldsymbol{\Sigma}}\end{pmatrix}\begin{pmatrix}\boldsymbol{V}^T \\ \widetilde{\boldsymbol{V}}^T\end{pmatrix} + \widetilde{\boldsymbol{Z}}\boldsymbol{B}^T.\end{aligned} \tag{28}$$

Here, $\widetilde{\boldsymbol{Z}} = \boldsymbol{Z} + \tau\boldsymbol{W}\boldsymbol{B} \in \mathbb{R}^{N_x \times 3}$ is arbitrary. The matrix $\widetilde{\boldsymbol{W}} = \boldsymbol{W}\boldsymbol{\Pi}_{\boldsymbol{B}}^{\perp}$, with compact SVD $\widetilde{\boldsymbol{W}} = \widetilde{\boldsymbol{U}}\widetilde{\boldsymbol{\Sigma}}\widetilde{\boldsymbol{V}}^T$, satisfies

$$\boldsymbol{U}^T\boldsymbol{W} = \boldsymbol{0} \implies \boldsymbol{U}^T\widetilde{\boldsymbol{U}} = \boldsymbol{0}, \tag{29a}$$

$$\boldsymbol{W}\boldsymbol{V} = \boldsymbol{0} \implies \widetilde{\boldsymbol{V}}^T\boldsymbol{V} = \boldsymbol{0}, \tag{29b}$$

$$\boldsymbol{\Pi}_{\boldsymbol{B}}^{\perp}\boldsymbol{B} = \boldsymbol{0} \implies \widetilde{\boldsymbol{V}}^T\boldsymbol{B} = \boldsymbol{0}, \tag{29c}$$

$$\left\|\boldsymbol{W}^T\boldsymbol{x}\right\|_2^2 = \left\|(\boldsymbol{I}-\boldsymbol{B}\boldsymbol{B}^T)\boldsymbol{W}^T\boldsymbol{x}\right\|_2^2 + \left\|\boldsymbol{B}\boldsymbol{B}^T\boldsymbol{W}^T\boldsymbol{x}\right\|_2^2 \geqslant \left\|\widetilde{\boldsymbol{W}}^T\boldsymbol{x}\right\|_2^2 \implies \left\|\widetilde{\boldsymbol{W}}\right\|_2 \leqslant 1. \tag{29d}$$

Therefore, such a decomposition exists. The equation

$$\frac{\eta}{a_2+\eta}\boldsymbol{M}\boldsymbol{\Pi}_{\boldsymbol{B}}^{\perp} = \begin{pmatrix}\boldsymbol{U} & \widetilde{\boldsymbol{U}}\end{pmatrix}\begin{pmatrix}\boldsymbol{\Sigma}+\tau & \boldsymbol{0} \\ \boldsymbol{0} & \tau\widetilde{\boldsymbol{\Sigma}}\end{pmatrix}\begin{pmatrix}\boldsymbol{V}^T \\ \widetilde{\boldsymbol{V}}^T\end{pmatrix} \tag{30}$$

gives a compact SVD. Using the fact that $\left\|\widetilde{\boldsymbol{\Sigma}}\right\|_2 = \left\|\widetilde{\boldsymbol{W}}\right\|_2 \leqslant 1$, we arrive at equation (25). □

Remark 3 (Efficient computation of the threshold)**.** *Although equation* (25) *defines* $\boldsymbol{Y}$ *implicitly, the threshold* τ *can be computed efficiently. Suppose we have computed the SVD* $\frac{\eta}{a_2+\eta}\boldsymbol{M}(\boldsymbol{I} - \boldsymbol{B}\boldsymbol{B}^T) = \widehat{\boldsymbol{U}}\widehat{\boldsymbol{\Sigma}}\widehat{\boldsymbol{V}}^T$ *with* $\widehat{\boldsymbol{\Sigma}} = \operatorname{diag}(\sigma_1, \cdots, \sigma_n)$*. Then* τ *satisfies*

$$0 = (a_2+\eta)\tau - 2\sum_{k=1}^{n}\max\left\{\sigma_k - \tau, 0\right\}. \tag{31}$$

The right-hand side is a convex, monotonically increasing, piecewise linear function of τ with breakpoints at $\sigma_1, \ldots, \sigma_n$. The root can be located by binary search over the breakpoints in $\mathcal{O}(\log n)$ comparisons, followed by an explicit solve within the linear piece, for a total cost of $\mathcal{O}(n \log n)$ floating point operations (dominated by sorting the singular values). The SVD itself is only computed once and is the dominant computational cost. A partial SVD, computing only the singular values larger than τ, could be used, but may need to be later expanded since the value of τ is not known in advance, so it is not particularly efficient.

4.3 Restarted dual FISTA for the convex problem

The Douglas–Rachford splitting works directly on the primal problem, but requires computing the proximal operator of g at each iteration, which involves a full SVD (cf. Theorem 1 and Remark 3). An alternative is to work on the dual problem, where the structure of the conjugate functions may lead to cheaper iterations. Since f and g are both strongly convex (with parameters a_1 and a_2 respectively), their convex conjugates f^* and g^* have Lipschitz continuous gradients, making the dual problem amenable to gradient-based methods.

The FISTA algorithm [2] uses a Nesterov-type [21] momentum method to accelerate the convergence of proximal gradient descent from $\mathcal{O}(1/k)$ to $\mathcal{O}(1/k^2)$, where k is the iteration number. FISTA can be further accelerated by adaptive restart strategies [23, 1]. The approach of applying restarted FISTA to a dual problem for enforcing nonnegativity constraints was studied in [6]. We apply the same idea to the dual problem of the splitting formulation (19),

$$\begin{aligned} &\min_{\boldsymbol{\Lambda}} F(\boldsymbol{\Lambda}), \\ &\text{where } F(\boldsymbol{\Lambda}) = \left\{ f^*(-\boldsymbol{\Lambda}) + g^*(\boldsymbol{\Lambda}) \right\}, \end{aligned} \tag{32}$$

where f^* and g^* are convex conjugates of f and g respectively, and $\boldsymbol{\Lambda} \in \mathbb{R}^{m \times n}$ is the dual variable. Here, we require $a_1 > 0$ and $a_2 > 0$.

Since f is a_1-strongly convex and g is a_2-strongly convex, the conjugate functions f^* and g^* are differentiable with $\frac{1}{a_1}$-Lipschitz and $\frac{1}{a_2}$-Lipschitz gradients, respectively. In particular, the dual cost function $F(\boldsymbol{\Lambda}) = f^*(-\boldsymbol{\Lambda}) + g^*(\boldsymbol{\Lambda})$ takes finite values and has Lipschitz continuous gradients.

We first write down f^*:

$$\begin{aligned} f^*(\boldsymbol{\Lambda}) &= \sup_{\boldsymbol{X} \succcurlyeq -\boldsymbol{A}} \langle \boldsymbol{\Lambda}, \boldsymbol{X} \rangle - \frac{a_1}{2} \|\boldsymbol{X}\|_F^2 \\ &= \left(\langle \boldsymbol{\Lambda}, \boldsymbol{X} \rangle - \frac{a_1}{2} \|\boldsymbol{X}\|_F^2 \right) \Big|_{\boldsymbol{X} = \max\{-\boldsymbol{A}, a_1^{-1}\boldsymbol{\Lambda}\}}, \end{aligned} \tag{33}$$

since the problem is separable, and using a similar procedure as the derivation of $\operatorname{prox}_{\eta^{-1} g}$, we have

$$g^*(\boldsymbol{\Lambda}) = \left(\langle \widetilde{\boldsymbol{\sigma}}, \boldsymbol{\sigma} \rangle - \|\boldsymbol{\sigma}\|_{\ell^1}^2 - \frac{a_2}{2} \|\boldsymbol{\sigma}\|_{\ell^2}^2 \right) \Big|_{\boldsymbol{\sigma} = D_{\frac{2\|\boldsymbol{\sigma}\|_{\ell^1}}{a_2}}\left(\frac{\widetilde{\boldsymbol{\sigma}}}{a_2}\right)}, \tag{34}$$

where $\widetilde{\boldsymbol{\sigma}}$ is the singular value vector of $\boldsymbol{\Lambda} \Pi_{\boldsymbol{B}}^{\perp}$ and $\|\boldsymbol{\sigma}\|_{\ell^1}$ is the sum over all the absolute values of the entries of $\boldsymbol{\sigma}$.

Since both f^* and g^* are smooth, the proximal-gradient (FISTA) iteration can be set up in two symmetric ways: take the gradient of one conjugate and the proximal operator of the other. We describe both and call them *variant I* (gradient of f^*, proximal operator of g^*) and *variant II* (gradient of g^*, proximal operator of f^*).

Variant I (gradient of f^*, proximal operator of g^*). This variant requires the proximal operator of g^*. Since g is a_2-strongly convex, g^* has $\frac{1}{a_2}$-Lipschitz gradient, and $\operatorname{prox}_{\eta_k g^*}$ can be evaluated using the Moreau decomposition $\operatorname{prox}_{\eta_k g^*}(\boldsymbol{Z}) = \boldsymbol{Z} - \eta_k \operatorname{prox}_{\eta_k^{-1} g}(\eta_k^{-1}\boldsymbol{Z})$, where $\operatorname{prox}_{\eta_k^{-1} g}$ is the proximal operator computed in Theorem 1. The iteration is

$$\boldsymbol{\Lambda}_{k+1} = \operatorname{prox}_{\eta_k g^*}\left(\boldsymbol{\Theta}_k + \eta_k \nabla f^*(-\boldsymbol{\Theta}_k)\right), \tag{35a}$$

$$t_{k+1} = \frac{1+\sqrt{1+4t_k^2}}{2}, \tag{35b}$$

$$\boldsymbol{\Theta}_{k+1} = \boldsymbol{\Lambda}_{k+1} + \frac{t_k - 1}{t_{k+1}}(\boldsymbol{\Lambda}_{k+1} - \boldsymbol{\Lambda}_k), \tag{35c}$$

where $\boldsymbol{\Lambda}_k$ converges fast to $\boldsymbol{\Lambda}$, and the initial condition is taken to be $\boldsymbol{\Theta}_0 = \boldsymbol{\Lambda}_0$ and $t_0 = 1$. We require $\eta_k \leqslant \frac{1}{L_{f^*}} = a_1$ for stability.

Since this algorithm operates in the dual space, we need to recover the primal variable $\boldsymbol{X}$ from $\boldsymbol{\Lambda}$. We use $\boldsymbol{X} = \nabla f^*(-\boldsymbol{\Lambda})$, which always satisfies the nonnegativity condition $\boldsymbol{X} \geqslant -\boldsymbol{A}$ (but not necessarily the orthogonality constraint since the iteration never *exactly* reaches optimality). By the Fenchel–Young equality,

$$\langle \boldsymbol{\Lambda}, \boldsymbol{X} \rangle = f(\boldsymbol{X}) + f^*(\boldsymbol{\Lambda}) \iff \boldsymbol{X} \in \partial f^*(\boldsymbol{\Lambda}),$$

the primal variable is given by

$$\boldsymbol{X} = \nabla f^*(-\boldsymbol{\Lambda}) = \max\{-\boldsymbol{A}, -a_1^{-1}\boldsymbol{\Lambda}\}. \tag{36}$$

Variant II (gradient of g^*, proximal operator of f^*). Here we instead take the gradient of g^* and the proximal operator of f^*. The iteration is

$$\boldsymbol{\Lambda}_{k+1} = \operatorname{prox}_{\eta_k\,[f^*(-\,\cdot\,)]}\left(\boldsymbol{\Theta}_k - \eta_k \nabla g^*(\boldsymbol{\Theta}_k)\right), \tag{37a}$$

$$t_{k+1} = \tfrac{1}{2}\left(1+\sqrt{1+4t_k^2}\right), \quad \boldsymbol{\Theta}_{k+1} = \boldsymbol{\Lambda}_{k+1} + \tfrac{t_k-1}{t_{k+1}}(\boldsymbol{\Lambda}_{k+1} - \boldsymbol{\Lambda}_k), \tag{37b}$$

with step size $\eta_k \leqslant \frac{1}{L_{g^*}} = a_2$. The gradient $\nabla g^*(\boldsymbol{\Lambda}) = D_{\frac{2\|\boldsymbol{X}\|_*}{a_2}}(a_2^{-1}\boldsymbol{\Lambda}\boldsymbol{\Pi}_{\boldsymbol{B}}^{\perp})$ requires an SVD, while the proximal operator of f^* is now the cheap closed-form map. By the Moreau decomposition,

$$\operatorname{prox}_{\eta\,[f^*(-\,\cdot\,)]}(\boldsymbol{Z}) = \boldsymbol{Z} + \eta \operatorname{prox}_{\eta^{-1} f}(-\eta^{-1}\boldsymbol{Z}) = \boldsymbol{Z} + \eta \max\left\{-\boldsymbol{A},\ -\tfrac{1}{a_1+\eta}\boldsymbol{Z}\right\}. \tag{38}$$

The recovered primal variable is now $\boldsymbol{X} = \nabla g^*(\boldsymbol{\Lambda}) = D_{\frac{2\|\boldsymbol{X}\|_*}{a_2}}(a_2^{-1}\boldsymbol{\Lambda}\boldsymbol{\Pi}_{\boldsymbol{B}}^{\perp})$, which *exactly* satisfies the orthogonality constraint $\boldsymbol{X}\boldsymbol{B} = \boldsymbol{0}$ and is low-rank, but is not necessarily nonnegative until convergence (the mirror image of variant I, whose iterate is always nonnegative but only orthogonal at convergence).

Remark 4 (Which variant to use)**.** *Both variants converge to the same global minimizer of* (2) *and, as Table 1 shows, have comparable per-iteration cost (each needs one SVD per iteration) and nearly the same convergence rate, with variant I slightly faster and cheaper (its primal recovery* (36) *is an elementwise clip rather than an SVD). They differ in which constraint the iterate satisfies exactly before convergence: variant I keeps the nonnegativity $\boldsymbol{A} + \boldsymbol{X} \geqslant \boldsymbol{0}$, whereas variant II keeps the*

orthogonality $\boldsymbol{X}\boldsymbol{B} = \mathbf{0}$. *Since the entire purpose of the correction is to recover nonnegativity, we recommend variant I: it returns a nonnegative iterate at every step and can therefore be stopped early while still yielding a physically admissible (nonnegative) solution, whereas variant II may return a solution with small negative entries until it has fully converged.*

To eliminate the oscillation caused by the momentum term, restart strategies are employed. For strongly convex problems, a fixed restart with an optimal period produces linear convergence [1]. Since our dual problem (32) is not strongly convex, we also consider adaptive restart methods [22, 23]. The function-value-based and gradient-based restart criteria are

$$F(\boldsymbol{\Lambda}_k) > F(\boldsymbol{\Lambda}_{k-1}), \tag{39}$$

and

$$\langle \boldsymbol{G}(\boldsymbol{\Theta}_{k-1}),\ \boldsymbol{\Lambda}_k - \boldsymbol{\Lambda}_{k-1} \rangle > 0, \tag{40a}$$

where

$$\boldsymbol{G}(\boldsymbol{\Theta}_{k-1}) = \frac{1}{\eta_k} \left(\boldsymbol{\Theta}_{k-1} - \boldsymbol{\Lambda}_k \right). \tag{40b}$$

4.4 Restarted dual accelerated gradient descent for the convex problem

If both a_1 and a_2 are positive, the dual cost function F in (32) has Lipschitz continuous gradient with Lipschitz constant $L = \frac{1}{a_1} + \frac{1}{a_2}$. In this case, we use a fully explicit accelerated gradient descent method without proximal operators. We apply Nesterov's accelerated gradient descent [21] to the dual problem and use the same restarting strategies as in the previous subsection.

The gradient of the dual cost function F is

$$\nabla F(\boldsymbol{\Lambda}) = -\nabla f^*(-\boldsymbol{\Lambda}) + \nabla g^*(\boldsymbol{\Lambda}), \tag{41}$$

where ∇f^* is given by (36) and ∇g^* is given by

$$\boldsymbol{X} = \nabla g^*(\boldsymbol{\Lambda}) = D_{\frac{2\|\boldsymbol{X}\|_*}{a_2}} \left(\frac{1}{a_2} \boldsymbol{\Lambda} \boldsymbol{\Pi}_{\boldsymbol{B}}^{\perp} \right). \tag{42}$$

The accelerated gradient iteration is

$$\boldsymbol{\Lambda}_{k+1} = \boldsymbol{\Theta}_k - \eta_k \left(\nabla g^*(\boldsymbol{\Theta}_k) - \nabla f^*(-\boldsymbol{\Theta}_k) \right), \tag{43}$$

$$t_{k+1} = \frac{1 + \sqrt{1 + 4t_k^2}}{2}, \tag{44}$$

$$\boldsymbol{\Theta}_{k+1} = \boldsymbol{\Lambda}_{k+1} + \frac{t_k - 1}{t_{k+1}} (\boldsymbol{\Lambda}_{k+1} - \boldsymbol{\Lambda}_k), \tag{45}$$

where $\boldsymbol{\Lambda}_k$ converges to $\boldsymbol{\Lambda}$, and the initial condition is $\boldsymbol{\Theta}_0 = \boldsymbol{\Lambda}_0$ and $t_0 = 1$. We require $\eta_k \leqslant \frac{1}{L_F} = \frac{a_1 a_2}{a_1 + a_2}$ for stability. Similar to the FISTA, our restarting strategies include fixed restart, function-value-based restart (39), and gradient-based restart (40).

4.5 Dual PR+ conjugate gradient for the convex problem

We next apply the nonlinear conjugate gradient method to the dual problem (32). We use the Polak–Ribière plus (PR+) variant [25] with an explicit step size formula from [27] to avoid line search. The algorithm is

$$\boldsymbol{G}_k = \nabla F(\boldsymbol{\Lambda}_k), \tag{46}$$

$$\beta_k = \begin{cases} 0, & k = 1, \\ \frac{\langle \boldsymbol{G}_k, \boldsymbol{G}_k - \boldsymbol{G}_{k-1} \rangle}{\|\boldsymbol{G}_{k-1}\|_F^2}, & k > 1, \end{cases} \tag{47}$$

$$\boldsymbol{D}_k = \beta_k \boldsymbol{D}_{k-1} - \boldsymbol{G}_k, \tag{48}$$

$$\alpha_k = -\frac{\delta \langle \boldsymbol{G}_k, \boldsymbol{D}_k \rangle}{\|\boldsymbol{D}_k\|_F^2} \tag{49}$$

$$\boldsymbol{\Lambda}_{k+1} = \boldsymbol{\Lambda}_k + \alpha_k \boldsymbol{D}_k. \tag{50}$$

Here, according to [27], we should require $\delta \in (0, \frac{1}{L_F})$ for stability.

4.6 Dual L-BFGS for the convex problem

The L-BFGS algorithm [19] is a quasi-Newton method that uses a limited number of past iterates to approximate the inverse Hessian, requiring only gradient evaluations. Applied to the dual problem (32), the algorithm is

$$\boldsymbol{G}_k = \nabla F(\boldsymbol{\Lambda}_k), \tag{51}$$

$$\boldsymbol{D}_k = -\boldsymbol{H}_k \boldsymbol{G}_k, \tag{52}$$

$$\alpha_k = \arg\min_{\alpha \geqslant 0} F(\boldsymbol{\Lambda}_k + \alpha \boldsymbol{D}_k), \tag{53}$$

$$\boldsymbol{\Lambda}_{k+1} = \boldsymbol{\Lambda}_k + \alpha_k \boldsymbol{D}_k, \tag{54}$$

$$\begin{aligned} \boldsymbol{H}_{k+1} = &\left(\prod_{j=k-p+1}^{k} \boldsymbol{J}_j \right)^T \boldsymbol{H}_0^k \prod_{j=k-p+1}^{k} \boldsymbol{J}_j \\ &+ \sum_{i=1}^{p} \gamma_{k-p+i} \left(\prod_{j=k-p+i+1}^{k} \boldsymbol{J}_j \right)^T \boldsymbol{s}_{k-p+i} \boldsymbol{s}_{k-p+i}^T \prod_{j=k-p+i+1}^{k} \boldsymbol{J}_j. \end{aligned} \tag{55}$$

Here, the iterates are assumed to be vectors, $\boldsymbol{J}_k = \boldsymbol{I} - \gamma_k \boldsymbol{y}_k \boldsymbol{s}_k^T$, $\boldsymbol{y}_j = \boldsymbol{G}_{j+1} - \boldsymbol{G}_j$, $\boldsymbol{s}_j = \boldsymbol{\Lambda}_{j+1} - \boldsymbol{\Lambda}_j$, $\gamma_j = \frac{1}{\boldsymbol{y}_j^T \boldsymbol{s}_j}$, and p is the number of history iterates to be stored. The matrix-vector product of $\boldsymbol{H}_k$ can be efficiently implemented through two iterations presented in [19]. The exact line search can be replaced by a Wolfe-type inexact search.

4.7 Tangent-space accelerated alternating projection for the non-convex problem

For the rank-constrained formulation (17), we develop an alternating projection algorithm that uses the manifold structure to avoid full SVD computations. We first introduce the relevant manifold and its projection properties.

Definition 3. *Let* $M_{r,\boldsymbol{B}} = \left\{ \boldsymbol{X} \in \mathbb{R}^{m\times n} \middle| \operatorname{rank}(\boldsymbol{X}) \leqslant r,\ \boldsymbol{X}\boldsymbol{B} = \boldsymbol{0} \right\}$ *denote the set of rank-at-most-r matrices satisfying the orthogonality constraint.*

Lemma 2. *The Frobenius-norm projection onto* $M_{r,\boldsymbol{B}}$ *is*

$$P_{M_{r,\boldsymbol{B}}}(\boldsymbol{Y}) = P_{M_r}\left(\boldsymbol{Y}\boldsymbol{\Pi}_{\boldsymbol{B}}^{\perp}\right), \tag{56}$$

where $P_{M_r}(\cdot)$ *is the standard rank-r SVD truncation.*

Proof. Let $\boldsymbol{Y}\boldsymbol{\Pi}_{\boldsymbol{B}}^{\perp} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$ be a full SVD. For any $\boldsymbol{Z} \in M_{r,\boldsymbol{B}}$,

$$\begin{aligned}
\|\boldsymbol{Y} - \boldsymbol{Z}\|_F^2 &= \|\boldsymbol{Y}\boldsymbol{\Pi}_{\boldsymbol{B}}\|_F^2 + \left\|\boldsymbol{Y}\boldsymbol{\Pi}_{\boldsymbol{B}}^{\perp} - \boldsymbol{Z}\right\|_F^2 \\
&= \|\boldsymbol{Y}\boldsymbol{\Pi}_{\boldsymbol{B}}\|_F^2 + \left\|\boldsymbol{Y}\boldsymbol{\Pi}_{\boldsymbol{B}}^{\perp}\right\|_F^2 - 2\operatorname{tr}\left(\boldsymbol{Z}^T\boldsymbol{Y}\boldsymbol{\Pi}_{\boldsymbol{B}}^{\perp}\right) + \|\boldsymbol{Z}\|_F^2 \\
&= \|\boldsymbol{Y}\boldsymbol{\Pi}_{\boldsymbol{B}}\|_F^2 + \left\|\boldsymbol{\Sigma} - \widehat{\boldsymbol{Z}}\right\|_F^2 \\
&\geqslant \|\boldsymbol{Y}\boldsymbol{\Pi}_{\boldsymbol{B}}\|_F^2 + \sum_{k=r+1}^{n} \Sigma_{k,k}^2,
\end{aligned} \tag{57}$$

where $\widehat{\boldsymbol{Z}} = \boldsymbol{U}^T\boldsymbol{Z}\boldsymbol{V}$ has rank at most r. The inequality is the Eckart–Young–Mirsky theorem: since $\Sigma_{k,k}$ are the singular values of $\boldsymbol{\Sigma}$, the best rank-r Frobenius approximation retains the largest r of them, with equality if and only if $\widehat{\boldsymbol{Z}} = P_{M_r}(\boldsymbol{\Sigma})$, i.e. $\boldsymbol{Z} = P_{M_r}\left(\boldsymbol{Y}\boldsymbol{\Pi}_{\boldsymbol{B}}^{\perp}\right)$. It is straightforward to check that $\boldsymbol{Z} = P_{M_r}\left(\boldsymbol{Y}\boldsymbol{\Pi}_{\boldsymbol{B}}^{\perp}\right) \in M_{r,\boldsymbol{B}}$. □

We first consider the alternating projection algorithm that projects onto the nonnegative set and onto $M_{r,\boldsymbol{B}}$, as shown in Algorithm 2.

Algorithm 2: Alternating projection (AP) for the non-convex problem

Input: $\boldsymbol{X}_0$, $\boldsymbol{A}$, $\boldsymbol{B}$, r, maxit
Output: $\boldsymbol{X}_{\text{maxit}+1}$

```
for k = 1 : maxit do
    Y_k = π_+(X_k) ;                          // element-wise
    X_{k+1} = P_{M_{r,B}}(Y_k) ;              // requires SVD
end
```

The expensive step is the projection $P_{M_{r,\boldsymbol{B}}}$, which requires a full SVD. Following [26], we replace this projection by first projecting onto the tangent space of $M_{r,\boldsymbol{B}}$ at the current iterate $\boldsymbol{X}_k$ and then truncating to rank r. This reduces the SVD cost from a full $m \times n$ SVD to a $2r \times 2r$ SVD. We now introduce the tangent-space machinery. In the following, we assume that the iterates $\boldsymbol{X}_k$ have exact rank r, so that the tangent space of the fixed-rank manifold is well-defined. This assumption holds generically in practice, especially since we choose r to be small.

Lemma 3. *The tangent space of the fixed-rank manifold* M_r *at* $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T \in M_r$ *(compact SVD*

form, with $\boldsymbol{\Sigma}$ having all positive diagonal entries) is

$$\begin{aligned}
&T_{M_r}(\boldsymbol{X}) \\
&= \left\{\boldsymbol{U}\boldsymbol{W}^T + \boldsymbol{Z}\boldsymbol{V}^T \middle| \boldsymbol{W} \in \mathbb{R}^{n\times r}, \boldsymbol{Z} \in \mathbb{R}^{m\times r}\right\} \\
&= \left\{\boldsymbol{U}\boldsymbol{M}\boldsymbol{V}^T + \boldsymbol{U}_p\boldsymbol{V}^T + \boldsymbol{U}\boldsymbol{V}_p^T \middle| \boldsymbol{U}_p^T\boldsymbol{U} = \boldsymbol{0}, \boldsymbol{V}_p^T\boldsymbol{V} = \boldsymbol{0}, \boldsymbol{M} \in \mathbb{R}^{r\times r}, \boldsymbol{U}_p \in \mathbb{R}^{m\times r}, \boldsymbol{V}_p \in \mathbb{R}^{n\times r}\right\} \\
&= \left\{ \begin{pmatrix}\boldsymbol{U} & \boldsymbol{U}^\perp\end{pmatrix} \begin{pmatrix}\boldsymbol{C}_1 & \boldsymbol{C}_2 \\ \boldsymbol{C}_3 & \boldsymbol{0}\end{pmatrix} \begin{pmatrix}\boldsymbol{V} & \boldsymbol{V}^\perp\end{pmatrix}^T \middle| \boldsymbol{C}_1 \in \mathbb{R}^{r\times r}, \boldsymbol{C}_2 \in \mathbb{R}^{r\times(n-r)}, \boldsymbol{C}_3 \in \mathbb{R}^{(m-r)\times r} \right\}.
\end{aligned} \tag{58}$$

Proof. See [35] and [29]. □

Lemma 4. *The least square projection (in Frobenius norm) onto $T_{M_r}(\boldsymbol{X})$ is*

$$P_{T_{M_r}(\boldsymbol{X})}(\boldsymbol{Y}) = \boldsymbol{\Pi}_{\boldsymbol{U}}\boldsymbol{Y} + \boldsymbol{Y}\boldsymbol{\Pi}_{\boldsymbol{V}} - \boldsymbol{\Pi}_{\boldsymbol{U}}\boldsymbol{Y}\boldsymbol{\Pi}_{\boldsymbol{V}}, \tag{59}$$

where $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$ is a rank-r SVD.

Proof. See [35] and [29]. □

Note that these computations can be done efficiently if the matrix multiplication is done in the right order; e.g., $\boldsymbol{\Pi}_{\boldsymbol{U}}\boldsymbol{Y} = \boldsymbol{U}\boldsymbol{U}^T\boldsymbol{Y}$ is calculated as $\boldsymbol{U}\left(\boldsymbol{U}^T\boldsymbol{Y}\right)$, and the final outer multiplication is not done explicitly but left in factored form (see Remark 5).

Lemma 5. *The tangent space of $M_{r,\boldsymbol{B}}$ at $\boldsymbol{X} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T \in M_{r,\boldsymbol{B}}$ (compact SVD form) is*

$$\begin{aligned}
&T_{M_{r,\boldsymbol{B}}}(\boldsymbol{X}) \\
&= \left\{\boldsymbol{U}\boldsymbol{W}^T + \boldsymbol{Z}\boldsymbol{V}^T \middle| \boldsymbol{W} \in \mathbb{R}^{n\times r}, \boldsymbol{Z} \in \mathbb{R}^{m\times r}, \boldsymbol{W}^T\boldsymbol{B} = \boldsymbol{0}\right\} \\
&= \left\{\boldsymbol{U}\boldsymbol{M}\boldsymbol{V}^T + \boldsymbol{U}_p\boldsymbol{V}^T + \boldsymbol{U}\boldsymbol{V}_p^T\boldsymbol{\Pi}_{\boldsymbol{B}}^{\perp} \middle| \boldsymbol{U}_p^T\boldsymbol{U} = \boldsymbol{0}, \boldsymbol{V}_p^T\boldsymbol{V} = \boldsymbol{0}, \boldsymbol{M} \in \mathbb{R}^{r\times r}, \boldsymbol{U}_p \in \mathbb{R}^{m\times r}, \boldsymbol{V}_p \in \mathbb{R}^{n\times r}\right\}.
\end{aligned} \tag{60}$$

Proof. Since $\boldsymbol{X} \in M_{r,\boldsymbol{B}}$, we have $\boldsymbol{V}^T\boldsymbol{B} = \boldsymbol{0}$. The orthogonality constraint $\boldsymbol{W}^T\boldsymbol{B} = \boldsymbol{0}$ follows from differentiating $\boldsymbol{X}\boldsymbol{B} = \boldsymbol{0}$ along the manifold. □

Lemma 6. *The least square projection (in Frobenius norm) onto $T_{M_r,\boldsymbol{B}}(\boldsymbol{X})$ is*

$$P_{T_{M_{r,\boldsymbol{B}}}(\boldsymbol{X})}(\boldsymbol{Y}) = P_{T_{M_r}(\boldsymbol{X})}(\boldsymbol{Y})\boldsymbol{\Pi}_{\boldsymbol{B}}^{\perp} = P_{T_{M_r}(\boldsymbol{X})}\left(\boldsymbol{Y}\boldsymbol{\Pi}_{\boldsymbol{B}}^{\perp}\right). \tag{61}$$

Proof. Since $\boldsymbol{X} \in M_{r,\boldsymbol{B}}$, we have $\boldsymbol{V}^T\boldsymbol{B} = \boldsymbol{0}$. As a result, there exists a matrix $\boldsymbol{K}$ such that the matrix $\begin{pmatrix}\boldsymbol{V} & \boldsymbol{B} & \boldsymbol{K}\end{pmatrix}$ is orthonormal. Using matrix $\boldsymbol{K}$, we can rewrite the tangent space as

$$\begin{aligned}
&T_{M_{r,\boldsymbol{B}}}(\boldsymbol{X}) \\
&= \left\{ \begin{pmatrix}\boldsymbol{U} & \boldsymbol{U}^\perp\end{pmatrix} \begin{pmatrix}\boldsymbol{C}_1 & \boldsymbol{0} & \boldsymbol{C}_2 \\ \boldsymbol{C}_3 & \boldsymbol{0} & \boldsymbol{0}\end{pmatrix} \begin{pmatrix}\boldsymbol{V} & \boldsymbol{B} & \boldsymbol{K}\end{pmatrix}^T \middle| \boldsymbol{C}_1 \in \mathbb{R}^{r\times r}, \boldsymbol{C}_2 \in \mathbb{R}^{r\times(n-r-3)}, \boldsymbol{C}_3 \in \mathbb{R}^{(m-r)\times r} \right\}.
\end{aligned} \tag{62}$$

Then, the least square projection is given by

$$\begin{aligned}
P_{T_{M_{r,\boldsymbol{B}}}(\boldsymbol{X})}(\boldsymbol{Y}) &= \boldsymbol{\Pi}_{\boldsymbol{U}}\boldsymbol{Y}\boldsymbol{\Pi}_{\boldsymbol{V}} + \boldsymbol{\Pi}_{\boldsymbol{U}}^{\perp}\boldsymbol{Y}\boldsymbol{\Pi}_{\boldsymbol{V}} + \boldsymbol{\Pi}_{\boldsymbol{U}}\boldsymbol{Y}\boldsymbol{\Pi}_{\boldsymbol{K}} \\
&= \boldsymbol{\Pi}_{\boldsymbol{U}}\boldsymbol{Y}\boldsymbol{\Pi}_{\boldsymbol{B}}^{\perp} + \boldsymbol{\Pi}_{\boldsymbol{U}}^{\perp}\boldsymbol{Y}\boldsymbol{\Pi}_{\boldsymbol{V}} \\
&= \boldsymbol{\Pi}_{\boldsymbol{U}}\boldsymbol{Y}\boldsymbol{\Pi}_{\boldsymbol{B}}^{\perp} + \boldsymbol{\Pi}_{\boldsymbol{U}}^{\perp}\boldsymbol{Y}\boldsymbol{\Pi}_{\boldsymbol{V}}\boldsymbol{\Pi}_{\boldsymbol{B}}^{\perp} = P_{T_{M_r}(\boldsymbol{X})}(\boldsymbol{Y})\boldsymbol{\Pi}_{\boldsymbol{B}}^{\perp}, \\
&= \boldsymbol{\Pi}_{\boldsymbol{U}}\boldsymbol{Y}\boldsymbol{\Pi}_{\boldsymbol{B}}^{\perp} + \boldsymbol{\Pi}_{\boldsymbol{U}}^{\perp}\boldsymbol{Y}\boldsymbol{\Pi}_{\boldsymbol{B}}^{\perp}\boldsymbol{\Pi}_{\boldsymbol{V}} = P_{T_{M_r}(\boldsymbol{X})}\left(\boldsymbol{Y}\boldsymbol{\Pi}_{\boldsymbol{B}}^{\perp}\right).
\end{aligned} \tag{63}$$

□

Remark 5. *Since* $\operatorname{rank}(P_{T_{M_{r,\boldsymbol{B}}}(\boldsymbol{X})}(\boldsymbol{Y})) \leqslant 2r$*, the tangent-space projection can be stored in a low-rank factored form.*

Using these results, we obtain the tangent-space accelerated alternating projection (TAP) algorithm in Algorithm 3. At each iteration, we project $\boldsymbol{Y}_k$ onto the tangent space of $M_{r,\boldsymbol{B}}$ at $\boldsymbol{X}_k$, and then truncate to rank r. The implementation of this combined step is given in Algorithm 4.

Algorithm 3: TAP for non-convex problem

Input: $\boldsymbol{X}_0$, $\boldsymbol{A}$, $\boldsymbol{B}$, r, maxit
Output: $\boldsymbol{X}_{\text{maxit}+1}$

for $k = 1 : \mathit{maxit}$ **do**
  $\boldsymbol{Y}_k = \pi_+(\boldsymbol{X}_k)$;
  $\boldsymbol{X}_{k+1} = P_{M_r}\left(P_{T_{M_{r,\boldsymbol{B}}}(\boldsymbol{X}_k)}(\boldsymbol{Y}_k)\right)$; `// rank` r`, and SVD also computed & stored`
end

Algorithm 4: TP operator $P_{M_r}\left(P_{T_{M_{r,\boldsymbol{B}}}(\boldsymbol{X})}(\boldsymbol{Y})\right)$ for non-convex problem

Input: $\boldsymbol{X}_k = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$ (rank-r compact SVD), $\boldsymbol{B}$, $\boldsymbol{Y}_k$, r
Output: $\boldsymbol{X}_{k+1}$ (in compact SVD form)

$\boldsymbol{Q}_1\boldsymbol{R}_1 = \boldsymbol{\Pi}_{\boldsymbol{U}}^{\perp}\boldsymbol{Y}_k\boldsymbol{V}$; `// compact QR`
$\boldsymbol{Q}_2\boldsymbol{R}_2 = \boldsymbol{\Pi}_{\boldsymbol{V}}^{\perp}\boldsymbol{Y}_k^T\boldsymbol{U}$; `// compact QR`
`// by orthogonality:` $\boldsymbol{U}^T\boldsymbol{Q}_1 = \boldsymbol{0}$ `and` $\boldsymbol{V}^T\boldsymbol{Q}_2 = \boldsymbol{0}$
$\widetilde{\boldsymbol{Q}}\widetilde{\boldsymbol{R}} = \boldsymbol{\Pi}_{\boldsymbol{B}}^{\perp}\begin{pmatrix}\boldsymbol{V} & \boldsymbol{Q}_2\end{pmatrix}$; `// compact QR`
$\boldsymbol{E} = \begin{pmatrix}\boldsymbol{U}^T\boldsymbol{Y}_k\boldsymbol{V} & \boldsymbol{R}_2 \\ \boldsymbol{R}_1 & \boldsymbol{0}\end{pmatrix}\widetilde{\boldsymbol{R}}^T$; `//` $\boldsymbol{E}$ `is` $2r \times 2r$
$\widehat{\boldsymbol{U}}\widehat{\boldsymbol{\Sigma}}\widehat{\boldsymbol{V}}^T = \boldsymbol{E}$; `//` $2r \times 2r$ `SVD`
`// rank-`$2r$ `factorization:` $P_{T_{M_{r,\boldsymbol{B}}}(\boldsymbol{X}_k)}(\boldsymbol{Y}_k) = \begin{pmatrix}\boldsymbol{U} & \boldsymbol{Q}_1\end{pmatrix}\widehat{\boldsymbol{U}}\,\widehat{\boldsymbol{\Sigma}}\,\widehat{\boldsymbol{V}}^T\widetilde{\boldsymbol{Q}}^T$
Truncate to rank r: keep only the top r singular triplets of $\widehat{\boldsymbol{U}}\widehat{\boldsymbol{\Sigma}}\widehat{\boldsymbol{V}}^T$ to form $\boldsymbol{X}_{k+1}$

Remark 6 (Computational cost)**.** *All iterates* $\boldsymbol{X}_k$ *are stored in rank-r factored form* $\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$*. The computational cost per iteration of Algorithm 4 is* $\mathcal{O}(mnr+(m+n)r^2+r^3)$*, which is significantly less than the* $\mathcal{O}(mn\min(m,n))$ *cost of the full SVD required by the algorithms for the convex formulation.*

5 Numerical experiments

5.1 Test problem and setup

We test all algorithms on a 1D1V Landau damping problem discretized on an $N_x = 64$, $N_v = 128$ grid. The low-rank solution has $\operatorname{rank}(\boldsymbol{A}_1) = 3$ and $\operatorname{rank}(\boldsymbol{A}_2) = 26$. The input data are shown

in Figure 1. The scaled matrix $\boldsymbol{A}$ has a negativity measure of $\|\min\{\boldsymbol{A}, \boldsymbol{0}\}\|_F / \|\boldsymbol{A}\|_F = 3.08\%$, which provides a lower bound on the relative correction error: any nonnegative correction must have $\|\boldsymbol{X}\|_F / \|\boldsymbol{A}\|_F \geqslant 3.08\%$.

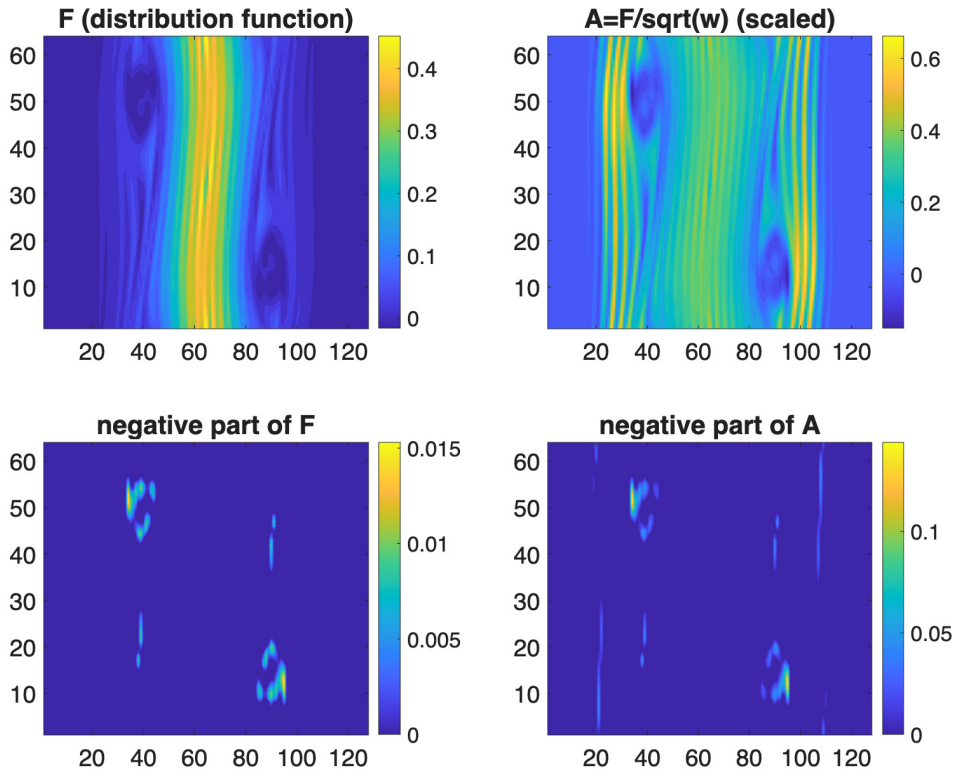


Figure 1: Input matrix $\boldsymbol{F}$, scaled matrix $\boldsymbol{A}$, and their negative parts.

For all algorithms applied to the convex formulation (2), after experimentation (see Fig. 2) we find that $a = 1000$ is a good compromise between the two objectives, and henceforth set $a = 1000$. We also set $a_1 = a_2 = a/2$, where a_1 and a_2 are the splitting parameters of the Douglas–Rachford decomposition (19), and note that the choice of splitting $a = a_1 + a_2$ does not produce visible differences in the results. Each convergence plot shows the relative correction $\|\boldsymbol{X}\|_F / \|\boldsymbol{A}\|_F$, the orthogonality violation $\|\boldsymbol{X}\boldsymbol{B}\|_F$, the rank of $\boldsymbol{X}$, and the relative change between successive iterates, for several values of the parameter a or the rank constraint r.

The converged relative correction $\|\boldsymbol{X}\|_F / \|\boldsymbol{A}\|_F$ and the rank of $\boldsymbol{X}$ depend on the penalty parameter a. Computed using the Douglas–Rachford splitting with 1000 iterations, as a increases through $1, 100, 1000, 10^4, 10^5$ the relative correction is $4.03\%, 3.31\%, 3.29\%, 3.29\%, 3.29\%$, decreasing monotonically toward the theoretical lower bound of 3.08%, while the rank of $\boldsymbol{X}$ is $38, 23, 33, 39, 45$; this reflects the trade-off between correction quality and low-rank structure controlled by a. In the

comparison below we fix $a = 1000$.

Table 1 compares all algorithms on the 64×128 data, reporting the relative correction $\|\boldsymbol{X}\|_F / \|\boldsymbol{A}\|_F$, the rank of $\boldsymbol{X}$, the orthogonality violation $\|\boldsymbol{X}\boldsymbol{B}\|_F$, the minimum of the corrected solution $\min(\boldsymbol{A}+\boldsymbol{X})$ (nonnegativity), the number of iterations to convergence, and the total CPU time. Each iterative method is run until the relative change of the iterate falls below 10^{-8} or a cap of 1000 iterations is reached. TAP uses rank $r = 20$. For reference we also include a naive low-rank baseline: take $\boldsymbol{X} = \max\{-\boldsymbol{A}, \boldsymbol{0}\}$ (which makes $\boldsymbol{A}+\boldsymbol{X}$ nonnegative), truncate it to rank r by an SVD, and project the result onto $\boldsymbol{\Pi}_{\boldsymbol{B}}^{\perp}$ to enforce $\boldsymbol{X}\boldsymbol{B} = \boldsymbol{0}$. This is a direct, non-iterative construction. The baseline is essentially free and satisfies the rank and orthogonality constraints exactly, but it does not enforce nonnegativity, so $\min(\boldsymbol{A}+\boldsymbol{X}) = -1.7 \times 10^{-2} < 0$. All convex algorithms converge to the same global minimizer (relative correction 3.29%, rank 33). The methods that recover the primal iterate as $\boldsymbol{X} = \nabla f^*(-\boldsymbol{\Lambda})$ (DR, dual FISTA I, dual AGD, dual PR+ CG, and dual L-BFGS) keep $\min(\boldsymbol{A}+\boldsymbol{X}) = 0$ exactly at every iterate and satisfy the orthogonality constraint only in the limit. Conversely, dual FISTA II recovers $\boldsymbol{X} = \nabla g^*(\boldsymbol{\Lambda})$, which is exactly orthogonal and low rank ($\|\boldsymbol{X}\boldsymbol{B}\|_F$ at machine precision) but only nonnegative in the limit (see Section 4.3). The two FISTA variants converge at nearly the same rate, with variant I slightly faster and cheaper per iteration. Douglas–Rachford splitting reaches the minimizer in the fewest iterations, whereas the dual PR+ conjugate gradient is the slowest and does not meet the tolerance within 1000 iterations (its iterate has not collapsed to rank 33 and its orthogonality violation remains 6×10^{-9}). Although L-BFGS converges in relatively few iterations, its line search evaluates the objective several times per iteration, so its total time is larger. TAP attains a comparable correction with exact orthogonality and machine-precision nonnegativity at a lower rank and the lowest cost among the iterative methods.

method	$\|\boldsymbol{X}\|_F / \|\boldsymbol{A}\|_F$	$\operatorname{rank}(\boldsymbol{X})$	$\|\boldsymbol{X}\boldsymbol{B}\|_F$	$\min(\boldsymbol{A}+\boldsymbol{X})$	iters	time (s)
DR splitting	3.29%	33	7.4×10^{-10}	0	65	0.10
dual FISTA I	3.29%	33	3.2×10^{-12}	0	201	0.28
dual FISTA II	3.29%	33	1.4×10^{-16}	-1.9×10^{-16}	214	0.39
dual AGD	3.29%	33	3.8×10^{-12}	0	308	0.33
dual PR+ CG	3.29%	47	6.4×10^{-9}	0	1000	1.08
dual L-BFGS	3.29%	33	5.6×10^{-10}	0	150	0.55
TAP	3.31%	20	5.2×10^{-17}	-1.3×10^{-16}	45	0.04
baseline	2.93%	20	7.4×10^{-17}	-1.7×10^{-2}	1	0.001

Table 1: Comparison of all algorithms on the 64×128 Landau data: relative correction, rank, orthogonality violation $\|\boldsymbol{X}\boldsymbol{B}\|_F$, nonnegativity $\min(\boldsymbol{A}+\boldsymbol{X})$, iterations to convergence, and total CPU time. Convex methods use $a = 1000$, the DR step size $\eta = 7a$ and relaxation $\alpha = 0.88$, and TAP uses $r = 20$. Convergence is declared when the relative change of the iterate drops below 10^{-8}, capped at 1000 iterations, and the baseline is a direct one-shot construction (its "iterations" entry is 1). The convex methods reach the same minimizer. All of them except dual FISTA II enforce nonnegativity exactly ($\min(\boldsymbol{A}+\boldsymbol{X}) = 0$ at every iterate), whereas dual FISTA II instead enforces orthogonality exactly ($\boldsymbol{X}\boldsymbol{B} = \boldsymbol{0}$), as discussed in Section 4.3. PR+ conjugate gradient does not meet the tolerance within 1000 iterations. TAP and the baseline satisfy $\boldsymbol{X}\boldsymbol{B} = \boldsymbol{0}$ to machine precision by construction. The baseline is the fastest but violates nonnegativity, $\min(\boldsymbol{A}+\boldsymbol{X}) < 0$. CPU times are indicative and machine-dependent.

5.2 Results for the convex formulation

Douglas–Rachford splitting. We use step size $\eta = 7a$ and relaxation parameter $\alpha = 0.88$ (jointly tuned for the fastest convergence at $a = 1000$, Figure 3) in Algorithm 1, starting from $\boldsymbol{Z}_1 = \boldsymbol{0}$ and running 1000 iterations. The convergence curves for different values of a are shown in Figure 2(a).

Dual methods. The restarted dual FISTA, restarted dual accelerated gradient descent, PR+ conjugate gradient, and L-BFGS all have comparable per-iteration cost to the Douglas–Rachford splitting. For the FISTA and accelerated gradient methods, we use step size $\eta_k = a_1$ and test three restarting strategies: fixed restart (period 100), function-value-based restart, and gradient-based restart. Figure 2(b) shows the fixed-restart FISTA and compares its two dual variants (Section 4.3), and Figure 3 compares the convergence of all nine convex algorithms at the representative value $a = 1000$.

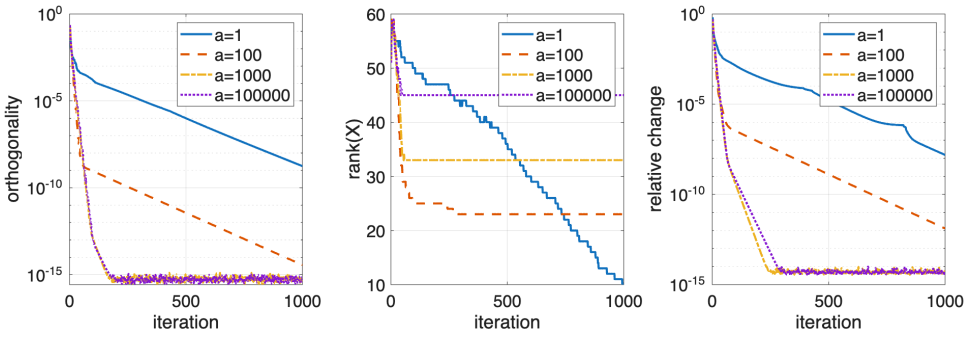
5.3 Results for the non-convex formulation

For the tangent-space accelerated alternating projection (TAP, Algorithm 3), we test different rank constraints r and run 200 iterations starting from $\boldsymbol{X}_0 = -\min\{\boldsymbol{A}, \boldsymbol{0}\}$. The convergence curves are shown in Figure 4.

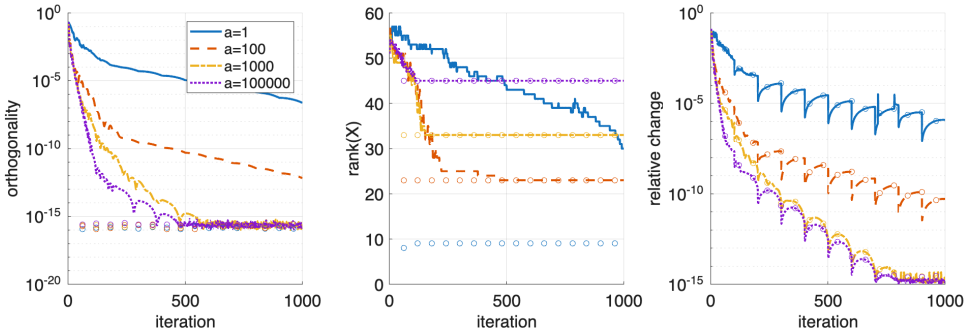
5.4 Empirical cost scaling with problem size

The timing experiments above use a single 64×128 matrix. To verify the cost scaling empirically, we run the conservative low-rank scheme of [13] for a strong Landau damping problem on a sequence of refined phase-space grids and measure the cost as the matrix size grows. We use the strong Landau damping initial condition $f_0(x, v) = \big(1 + \frac{1}{2}\cos\frac{x}{2}\big)\,\frac{1}{\sqrt{2\pi}} e^{-v^2/2}$ on $\Omega_x \times \Omega_v = [0, 4\pi] \times [-6, 6]$ and take the low-rank solution snapshot at $t = 10$, by which time the SVD-type truncation has introduced negative entries. For each grid we form the rescaled matrix $\boldsymbol{A}$ and its conservative orthogonal split $\boldsymbol{A} = \boldsymbol{A}_1 + \boldsymbol{A}_2$ with $\boldsymbol{A}_1 = \boldsymbol{A}\boldsymbol{B}\boldsymbol{B}^T$, exactly as in Sections 2.3–3.2. We use four grids with $N_v = 2N_x$, listed in Table 2. The orthogonality $\boldsymbol{A}_2\boldsymbol{B} = \boldsymbol{0}$ holds to machine precision ($\|\boldsymbol{A}_2\boldsymbol{B}\|_F \lesssim 10^{-13}$) on every grid. We emphasize that this strong Landau damping data (perturbation amplitude $\frac{1}{2}$) is different from the data used in the rest of the paper (Figure 1 and Table 1). We use it here because the same low-rank solver then generates a self-consistent family of nonnegative-correction problems across all grid resolutions, which is what we need to measure the cost scaling. Figure 5 shows the low-rank solution and its negative part for the 64×128 and 256×512 grids: the solution exhibits the same filamentary structure on both grids, while the truncation-induced negative entries become finer and smaller in magnitude as the grid is refined.

For convenience we write $m = N_x$ and $n = N_v$, so the low-rank field is an $m \times n$ matrix. The per-iteration cost of every convex algorithm is dominated by a single full $m \times n$ SVD, which costs $\mathcal{O}(mn \min(m, n))$, whereas a TAP iteration is dominated by one tangent-space projection (a $2r \times 2r$ SVD plus $\mathcal{O}(mnr)$ work, Algorithm 4) with a fixed rank r. Since $n = 2m$ here, we have $\min(m, n) = m$ and $mn = 2m^2$, so the convex per-iteration cost grows as $\mathcal{O}\big((mn)^{3/2}\big)$ while the TAP per-iteration cost grows only as $\mathcal{O}(mn)$. For each grid size we time both the dominant kernel and the full algorithm (Douglas–Rachford splitting for the convex problem, with $a = 10^3$, and TAP for the non-convex problem, with $r = 20$). To make the comparison fair we strip the convergence



(a) Douglas–Rachford splitting



(b) Dual FISTA with fixed restart, the two variants overlaid

Figure 2: Convergence of two convex algorithms for different values of a. (a) Douglas–Rachford splitting. (b) Dual FISTA with fixed restart, overlaying Variant I (lines, gradient of f^* and proximal operator of g^*) and Variant II (circles, gradient of g^* and proximal operator of f^*), with curves of the same color corresponding to the same value of a. Each row shows the orthogonality violation $\|\boldsymbol{X}\boldsymbol{B}\|_F$, the rank of $\boldsymbol{X}$, and the relative change between iterates. Variant II keeps the orthogonality violation at machine precision throughout, since its iterate satisfies $\boldsymbol{X}\boldsymbol{B} = \boldsymbol{0}$ exactly.

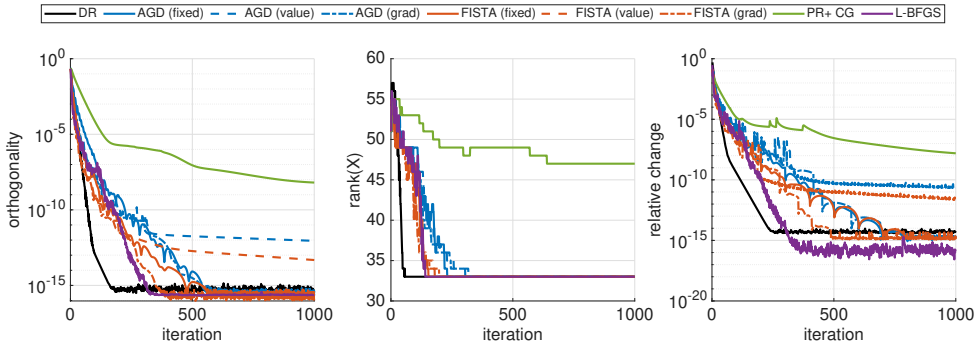


Figure 3: Convergence of all nine convex algorithms at the single penalty $a = 1000$ ($a_1 = a_2 = a/2$): the orthogonality violation $\|\boldsymbol{X}\boldsymbol{B}\|_F$, the rank of $\boldsymbol{X}$, and the relative change between iterates versus iteration. Color encodes the method family (Douglas–Rachford, accelerated gradient, FISTA, PR+ conjugate gradient, L-BFGS) and line style encodes the restart strategy (fixed, function-value, gradient), with dual FISTA shown for Variant I. The DR step size $\eta = 7a$ and relaxation $\alpha = 0.88$, jointly tuned at this a, give the fastest convergence (65 iterations, versus 114 at the untuned $\eta = 10^{0.5}a$, $\alpha = 0.9$). All methods reach the same minimizer (rank 33, relative correction 3.29%) except the PR+ conjugate gradient, which has not converged within 1000 iterations.

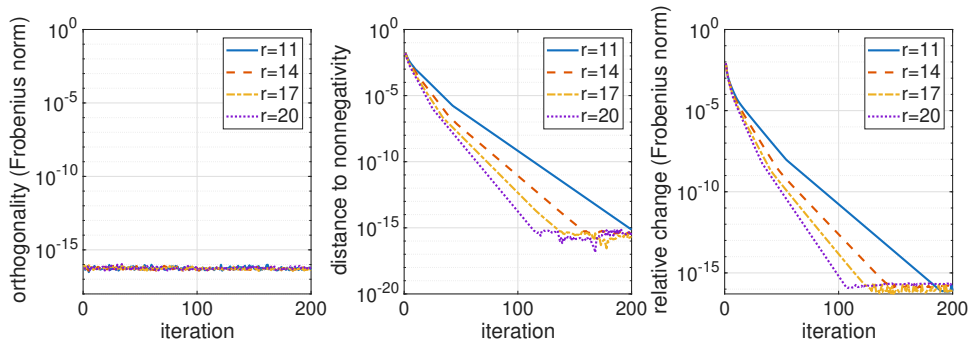


Figure 4: Tangent-space accelerated alternating projection (TAP): convergence for different rank constraints r. Left: orthogonality violation $\|\boldsymbol{X}\boldsymbol{B}\|_F$, which stays at machine precision throughout because TAP keeps $\boldsymbol{X}\boldsymbol{B} = \boldsymbol{0}$ exactly at every iterate. Middle: distance to the nonnegative set, $\max(-\min\{\boldsymbol{A} + \boldsymbol{X}, \boldsymbol{0}\})$. Right: relative change between successive iterates.

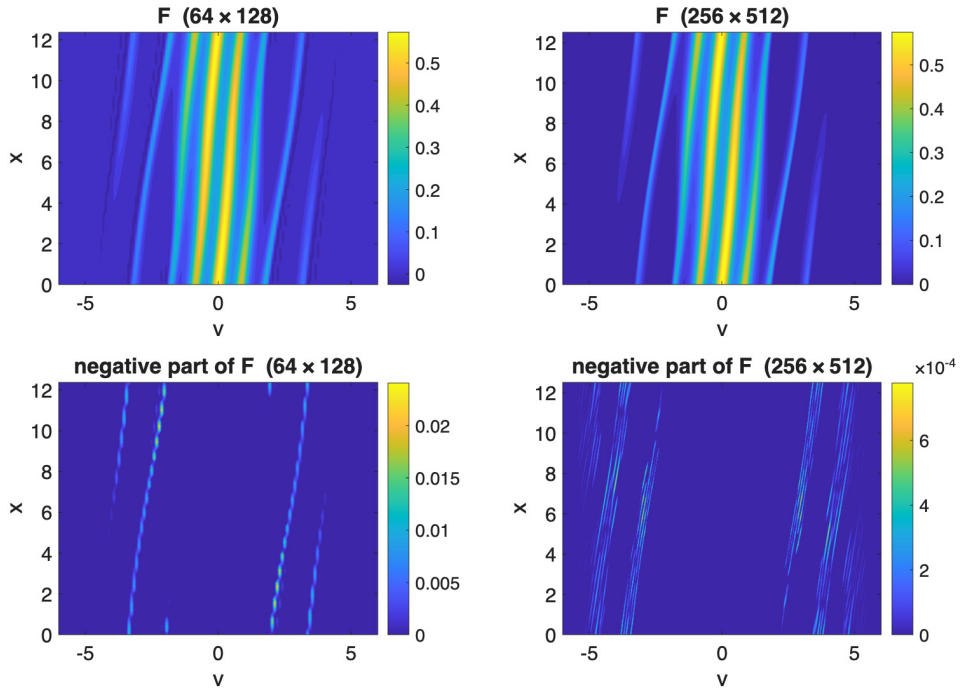


Figure 5: Low-rank Landau damping solution F (top) and its negative part $-\min\{F, \mathbf{0}\}$ (bottom) on the 64×128 (left) and 256×512 (right) grids, in the style of the left column of Figure 1. The negative entries, introduced by the SVD-type truncation, concentrate along the filament edges and become finer and smaller in magnitude under refinement.

		convex (DR splitting)		non-convex (TAP)		
grid $N_x \times N_v$	mn	full SVD (s)	DR / iter (s)	tangent proj. (s)	TAP / iter (s)	ratio
32×64	2,048	1.9×10^{-4}	2.6×10^{-4}	3.1×10^{-4}	3.0×10^{-4}	0.9
64×128	8,192	5.5×10^{-4}	8.9×10^{-4}	4.4×10^{-4}	4.9×10^{-4}	1.8
128×256	32,768	3.1×10^{-3}	4.1×10^{-3}	6.3×10^{-4}	8.7×10^{-4}	4.7
256×512	131,072	1.7×10^{-2}	2.8×10^{-2}	1.1×10^{-3}	1.6×10^{-3}	18.0

Table 2: Per-iteration cost scaling across grid sizes for the Landau damping problem (median of three trials, convergence diagnostics excluded). For the convex problem we report the cost of one full $m \times n$ SVD (the dominant kernel) and the full per-iteration cost of the Douglas–Rachford splitting. For the non-convex problem we report the cost of one tangent-space projection (the $2r \times 2r$-SVD kernel) and the full per-iteration cost of TAP, with $r = 20$ fixed. The last column is the per-iteration ratio, DR / TAP. The negativity measure $\|\min\{\boldsymbol{A}, \mathbf{0}\}\|_F / \|\boldsymbol{A}\|_F$ ranges from 2.8% to 11.3% and $\operatorname{rank}(\boldsymbol{A}_2)$ from 23 to 41 across the four grids.

diagnostics from both inner loops, so the reported times reflect only the algorithmic work. The results are reported in Table 2 and Figure 6.

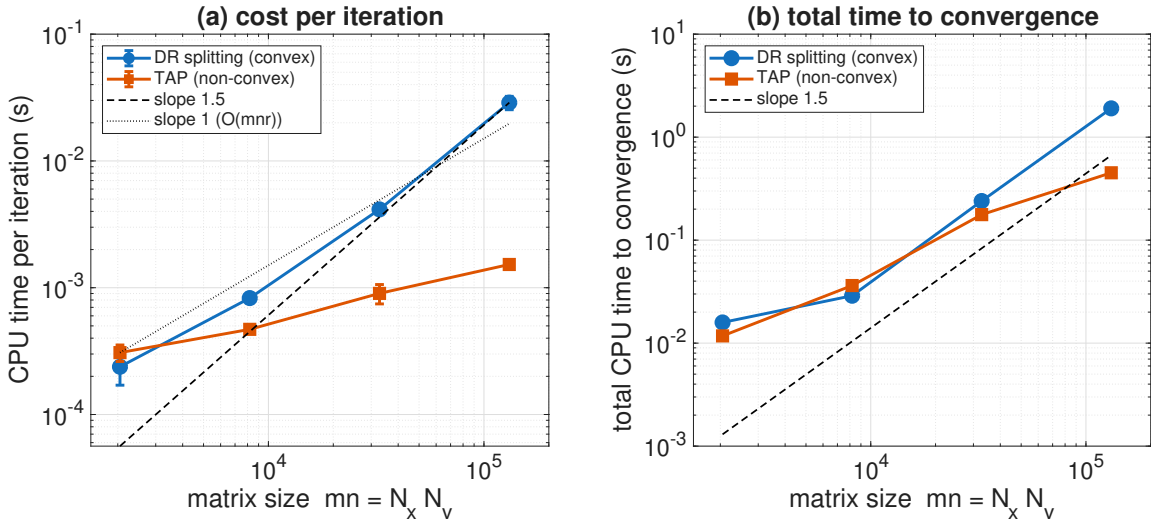


Figure 6: Cost scaling versus matrix size $mn = N_x N_v$ (log–log). (a) Per-iteration cost of the Douglas–Rachford splitting (convex) and the TAP algorithm (non-convex), plotted as the median over 20 runs with error bars of ± 1 standard deviation. The DR cost follows the predicted $(mn)^{3/2}$ slope at the larger sizes, while the TAP cost grows *sub-linearly* (empirical slope ≈ 0.4), below its asymptotic $\mathcal{O}(mnr)$ rate (slope 1). In this size range a TAP iteration is dominated by the size-independent cost of the tangent-space projection (the $2r \times 2r$ SVD and the QR of the tangent factors, $\mathcal{O}(r^3 + (m+n)r^2)$) rather than by the $\mathcal{O}(mnr)$ matrix products, which take only microseconds, so the linear regime would set in only at much larger mn. The per-iteration gap between DR and TAP nonetheless reaches about $18\times$ at 256×512. (b) Total wall-clock time to convergence (relative change of the iterate below 10^{-8}, with the DR step size tuned per grid): the two methods are comparable on the smaller grids, while TAP is about $4\times$ faster at 256×512. With its tuned step size DR needs fewer iterations than TAP, but its per-iteration full SVD makes each iteration much more expensive.

The measured costs confirm the predicted per-iteration scaling (Figure 6(a)). The convex per-iteration cost follows the $(mn)^{3/2}$ reference slope at the larger grids (the DR per-iteration time increases by a factor of about 6.9 over the last fourfold increase in mn, approaching the asymptotic factor of 8), while the TAP per-iteration cost grows much more slowly. The per-iteration advantage of TAP therefore grows monotonically with the problem size, from being slightly more expensive on the smallest 32×64 grid (where the highly optimized full SVD of a tiny matrix is cheaper than the TAP projection overhead) to being 18 times cheaper on the 256×512 grid.

The total cost is the per-iteration cost times the number of iterations to convergence, so we also examine how the iteration count scales with the problem size. We run both algorithms until the relative change of the iterate $\|\boldsymbol{X}_k - \boldsymbol{X}_{k-1}\|_F / \|\boldsymbol{X}_{k-1}\|_F$ falls below 10^{-8}, with $a = 10^3$ and $r = 20$ for TAP. For DR we tune the step size at each grid, so that DR is compared at its best at every

size. The optimal step grows with the problem (from $\eta \approx 2a$ on the coarsest grid to $\eta \approx 11a$ on the finest, with relaxation α between 0.90 and 0.94). The two algorithms reach comparable relative corrections $\|\boldsymbol{X}\|_F / \|\boldsymbol{A}\|_F$ (last two columns of Table 3), so the comparison is fair. With its step size tuned, DR converges in markedly fewer iterations than TAP, and the gap widens with refinement (DR from 30 to 82 iterations versus TAP from 32 to 264, Table 3). Each DR iteration nevertheless requires a full SVD, whereas a TAP iteration requires only a $2r \times 2r$ SVD, so DR's per-iteration cost grows much faster (Figure 6(a)). The two effects offset in the total wall-clock time (Figure 6(b)), which is comparable for the two methods on the three smaller grids (DR is even slightly faster at 64×128) while TAP is about $4\times$ faster on the finest 256×512 grid. Since the per-iteration cost gap scales as $\min(m, n)/r = m/r$ while DR's iteration advantage grows only mildly, we expect TAP's total-time advantage to keep widening on still finer grids.

		iterations to conv.		total time (s)		$\|\boldsymbol{X}\|_F / \|\boldsymbol{A}\|_F$	
grid $N_x \times N_v$	mn	DR	TAP	DR	TAP	DR	TAP
32×64	2,048	30	32	0.016	0.012	11.6%	11.6%
64×128	8,192	31	74	0.029	0.036	5.8%	6.2%
128×256	32,768	55	196	0.24	0.18	8.1%	9.5%
256×512	131,072	82	264	1.90	0.45	2.8%	3.7%

Table 3: Iterations to convergence (relative change of the iterate below 10^{-8}) and total wall-clock time for DR splitting and TAP ($r = 20$) across grid sizes, with $a = 10^3$. The DR step size is tuned per grid (the optimum grows with the grid: $\eta/a = 2.0, 2.5, 7.1, 11.2$, relaxation $\alpha = 0.90$ on the two coarse grids and 0.94 on the two fine grids). With its step size tuned, DR converges in fewer iterations than TAP, increasingly so as the grid is refined. Each DR iteration nevertheless requires a full SVD, so its per-iteration cost grows faster, and the total time favors TAP only on the finest grid. The last two columns show that the two algorithms reach comparable relative corrections. CPU times are indicative and machine-dependent.

Remark 7 (Accelerating the per-iteration SVD)**.** *The per-iteration cost of the convex algorithms is dominated by a single full $m \times n$ SVD (Table 2), so it is natural to ask whether a cheaper low-rank factorization can replace it. A maxvol-based cross (CUR) approximation [10], which samples a few rows and columns without ever forming the full matrix, is slower than the built-in SVD at every size we tested (for instance about 8 times slower at 128×256). The reason is that it is designed to minimize the number of entry evaluations, an advantage only when the entries are expensive to compute, as for implicit functions or high-dimensional tensors, whereas our matrices are explicit and stored in memory, so a single level-3 BLAS SVD is faster.*

A randomized SVD [15] is more promising, but its benefit is strongly rank-dependent, and this is the essential point. Its cost scales as $\mathcal{O}(mnr)$, so it beats the full SVD, which costs $\mathcal{O}(mn \min(m, n))$, only when the numerical rank r is a small fraction of $\min(m, n)$. This holds for the data matrix $\boldsymbol{A}$, which is genuinely low rank: there the randomized SVD reproduces the same singular triplets and is faster on the finer grids, by about three times on the finest (Table 4, upper block). It does not *hold for the matrices that the DR iteration actually factors. Each iteration applies the SVD not to $\boldsymbol{A}$ but to a proximal input $\boldsymbol{W}$, a dense intermediate matrix formed from the current iterates that is numerically full rank, and the soft-thresholded output retains a large fraction of the spectrum,*

roughly half of $\min(m,n)$ *(Table 4, lower block). On these matrices the same randomized SVD gives no speedup and is in fact slower at every size. Replacing the SVD by a randomized one therefore does not accelerate DR, even though it reproduces the same iterates to controllable accuracy. The low* $\mathcal{O}(mnr)$ *per-iteration cost is instead achieved by the tangent-space method, which keeps the iterate in rank-r factored form and never forms or factors a full-rank matrix.*

grid $N_x \times N_v$	r	full SVD (s)	randomized SVD (s)	speedup
data matrix $\boldsymbol{A}$ (genuinely low rank)				
64×128	31	2.9×10^{-4}	6.6×10^{-4}	$0.44\times$
128×256	43	2.2×10^{-3}	1.6×10^{-3}	$1.35\times$
256×512	44	9.5×10^{-3}	3.2×10^{-3}	$3.02\times$
DR proximal matrix $\boldsymbol{W}$ (numerically full rank)				
64×128	58	3.6×10^{-4}	1.3×10^{-3}	$0.29\times$
128×256	84	2.2×10^{-3}	3.9×10^{-3}	$0.58\times$
256×512	121	1.2×10^{-2}	1.3×10^{-2}	$0.95\times$

Table 4: Rank-dependence of the randomized-SVD speedup. The same randomized SVD (subspace iteration with oversampling) is applied to two matrices at each grid: the data matrix $\boldsymbol{A}$, which is genuinely low rank, and a representative Douglas–Rachford proximal input $\boldsymbol{W}$, a dense intermediate matrix formed from the current iterates that is numerically full rank. Here r is the numerical rank of $\boldsymbol{A}$ and the retained rank of the thresholded $\boldsymbol{W}$, and the randomized SVD targets this r. The speedup is the full-SVD time divided by the randomized-SVD time (fastest of seven trials), so a value above one means the randomized SVD is faster. On the low-rank $\boldsymbol{A}$ it is faster on the finer grids, whereas on the full-rank $\boldsymbol{W}$ it is slower at every size, because its $\mathcal{O}(mnr)$ cost helps only when $r \ll \min(m,n)$. CPU times are indicative and machine-dependent.

5.5 Application as a positivity limiter in the LoMaC scheme

So far we have applied the correction to a single fixed matrix. We now use it as a *positivity limiter* inside the time-dependent LoMaC low-rank scheme of [14], which conserves total mass, momentum *and* energy at the discrete level. As the low-rank solution is advanced, the SVD-type truncation at each step introduces small negative entries. We apply the correction periodically (every k steps and on the final step, each application an SVD-scale computation) to remove the unphysical negative entries while preserving the macroscopic moments, since its constraint $\boldsymbol{X}\boldsymbol{B} = \boldsymbol{0}$ leaves the three conserved densities (mass, momentum and energy) untouched. We compare two correction algorithms as the limiter. The first is the convex Douglas–Rachford splitting (DR, Algorithm 1), which imposes *no a-priori rank* on the correction: the corrected field takes whatever rank nonnegativity requires, and its rank is reduced only by the scheme's own conservative truncation $\mathcal{T}_{\varepsilon_{\mathrm{tr}}}$ at a tolerance $\varepsilon_{\mathrm{tr}}$. (For DR we use the step size $\eta = 10\,a$, found near-optimal by a step-size sweep on a representative correction subproblem: 83 iterations to drive $\|\boldsymbol{X}\boldsymbol{B}\|_F$ below 10^{-10}, versus 190 at $\eta = 10^{0.5}a$ and 602 at $\eta = a$.) The second is the rank-constrained TAP (Algorithm 3). The essential point is that its rank must *not* be fixed at some foreign value. Instead, we let it track the solution, taking $r = N + 3$ where N is the current numerical rank of the low-rank solution at each

application. This gives TAP the rank the field already carries, plus a small margin for the extra rank that nonnegativity introduces, so that, like DR, it is never artificially capped. (A fixed rank, e.g. $r = 20$, would cap the attainable nonnegativity, as we note below.)

We test the limiter on the bump-on-tail instability used to validate the LoMaC scheme (Example 5.4 of [14]), a well-resolved, genuinely low-rank run of the same family as the data in Figure 1, with initial datum

$$f_0(x, v) = (1 + \alpha \cos(kx))\Big(n_p\, e^{-v^2/2} + n_b\, e^{-(v-u)^2/(2v_t^2)}\Big),$$

$\alpha = 0.04$, $k = 0.3$, $n_p = \frac{9}{10\sqrt{2\pi}}$, $n_b = \frac{2}{10\sqrt{2\pi}}$, $u = 4.5$, $v_t = 0.5$, on $[0, 2\pi/k] \times [-13, 13]$, integrated to $T = 30$ with $\varepsilon = 10^{-4}$, using the validated LoMaC discretization (linear fifth-order upwind in x, CFL $= 0.1$ on both grids, so that the two runs share the same time-step criterion). We run two grids, 64×128 and 128×256, and apply the limiter every $k = 1000$ steps. Table 5 and Figure 7 summarize the results.

In this regime the LoMaC solution is genuinely low rank (numerical rank 28 on 64×128 and 49 on 128×256 at $T = 30$), and the scheme conserves total mass, momentum *and* energy to machine precision ($\sim 10^{-14}$ for all three). The SVD truncation nonetheless leaves a negativity $\|\min\{\boldsymbol{F}, \boldsymbol{0}\}\|_F \,/\, \|\boldsymbol{F}\|_F \approx 2\text{–}3 \times 10^{-4}$. The limiter removes it, and because the correction satisfies $\boldsymbol{X}\boldsymbol{B} = \boldsymbol{0}$ it does so without disturbing the three conserved quantities. Here the two correction algorithms differ in an instructive way (Table 5): TAP keeps $\boldsymbol{X}_k\boldsymbol{B} = \boldsymbol{0}$ *exactly at every iterate*, so the limited run conserves mass, momentum and energy to machine precision ($\sim 10^{-14}$), just like the unlimited LoMaC scheme. DR enforces $\boldsymbol{X}\boldsymbol{B} = \boldsymbol{0}$ only in the limit, so each application perturbs the moments by its orthogonality residual $\|\boldsymbol{X}\boldsymbol{B}\|_F$, and conservation is preserved only to that level ($\sim 10^{-10}$ at $\varepsilon_{\mathrm{tr}} = 10^{-4}$, improving to $\sim 10^{-12}$ as the correction is converged more tightly at $\varepsilon_{\mathrm{tr}} = 10^{-10}$). For exact conservation, then, the rank-constrained TAP limiter is preferable, which mirrors the constraint-preservation property noted in Section 5.6.

The other point concerns the interplay between nonnegativity and rank, and it explains why the stored solution is not nonnegative to machine zero by default. The correction itself enforces nonnegativity *absolutely*: the corrected field satisfies $\boldsymbol{A} + \boldsymbol{X} \geqslant \boldsymbol{0}$ to machine precision *before* truncation (for DR, $\boldsymbol{X}$ is the exact pointwise clip $\boldsymbol{X} = \max\{-\boldsymbol{A}, \cdot\}$, so $\boldsymbol{A} + \boldsymbol{X} \geqslant \boldsymbol{0}$ identically). Negativity reappears *only* when this corrected field is re-expressed in the rank-bounded low-rank format. A nonnegative field generically has higher rank than the smooth, slightly negative one, so the conservative truncation $\mathcal{T}_{\varepsilon_{\mathrm{tr}}}$ must discard a singular-value tail of norm $O(\varepsilon_{\mathrm{tr}})$. Because the corrected field sits exactly at zero where it had been negative, discarding this tail perturbs it by $O(\varepsilon_{\mathrm{tr}})$ and pushes those entries just below zero, leaving a negativity of magnitude $O(\varepsilon_{\mathrm{tr}})$. The negativity floor of the stored low-rank solution is therefore set by the truncation tolerance $\varepsilon_{\mathrm{tr}}$, not by the correction: at $\varepsilon_{\mathrm{tr}} = 10^{-4}$ (the scheme's own truncation level) the negativity is $\sim 2 \times 10^{-6}$ at a modestly higher rank ($49 \to 51$–53 on 128×256), while tightening to $\varepsilon_{\mathrm{tr}} = 10^{-10}$ drives it toward machine precision ($\sim 10^{-12}$, $\min \boldsymbol{F} \sim -10^{-11}$ to -10^{-12}) at a higher rank ($49 \to 84$ for DR, $49 \to 91$ for TAP). Provided the rank is allowed to grow, both algorithms reach comparable floors (Table 5 and Figure 7). This is also why letting TAP's rank track the solution ($r = N + 3$) matters: a fixed externally-imposed rank caps the attainable nonnegativity (with $r = 20$ the rank-20 correction stalls at $\min(\boldsymbol{A} + \boldsymbol{X}) \approx -1.4 \times 10^{-10}$ regardless of $\varepsilon_{\mathrm{tr}}$), whereas the adaptive rank, like the unconstrained convex correction, removes that cap. In short, the residual negativity is governed by how much rank one keeps in the low-rank format, not by the correction itself.

Figure 7(a) shows the negativity of the stored low-rank field (that is, after the correction *and* the re-truncation) just after each limiter application. The DR and TAP curves overlap at each $\varepsilon_{\mathrm{tr}}$,

reaching $\sim 2\times10^{-6}$ at $\varepsilon_{\mathrm{tr}}=10^{-4}$ and $\sim 10^{-12}$ at $\varepsilon_{\mathrm{tr}}=10^{-10}$, confirming that the floor follows $\varepsilon_{\mathrm{tr}}$ rather than the correction. (Between applications the per-step truncation at ε lets the negativity regrow, bounded by the no-limiter level, so a moderate k controls it while the final output, limited on the last step, is nonnegative to $\varepsilon_{\mathrm{tr}}$.) Figure 7(b) shows the rank: both limiters hold the solution at the higher rank that nonnegativity demands, increasingly so as $\varepsilon_{\mathrm{tr}}$ is tightened. The cost is modest and comparable for the two methods: with $k=1000$ the limiter performs 15 (64×128) or 30 (128×256) correction calls, raising the wall-clock time on 64×128 from 23 to 28 s and on 128×256 from 174 s to 208–230 s (a 20–30% overhead, dominated by the higher rank sustained between calls rather than by the correction solves). TAP is the cheaper of the two here, and its per-iteration advantage over DR grows on finer grids (Section 5.4).

mesh	limiter	mass dev.	mom. dev.	energy dev.	negativity	rank	CPU (s)
64×128	none	5.6×10^{-15}	2.5×10^{-14}	1.5×10^{-14}	1.8×10^{-4}	28	23
64×128	DR, $\varepsilon_{\mathrm{tr}}=10^{-4}$	4.4×10^{-14}	8.3×10^{-14}	1.6×10^{-13}	1.1×10^{-6}	27	28
64×128	TAP, $\varepsilon_{\mathrm{tr}}=10^{-4}$	6.6×10^{-15}	2.5×10^{-14}	1.6×10^{-14}	1.5×10^{-6}	28	28
64×128	DR, $\varepsilon_{\mathrm{tr}}=10^{-10}$	2.7×10^{-14}	2.2×10^{-13}	5.5×10^{-13}	2.1×10^{-12}	54	28
64×128	TAP, $\varepsilon_{\mathrm{tr}}=10^{-10}$	3.8×10^{-15}	2.6×10^{-14}	1.7×10^{-14}	2.8×10^{-12}	53	28
128×256	none	9.8×10^{-15}	4.0×10^{-14}	3.5×10^{-14}	3.0×10^{-4}	49	174
128×256	DR, $\varepsilon_{\mathrm{tr}}=10^{-4}$	1.7×10^{-11}	1.1×10^{-10}	6.3×10^{-11}	2.0×10^{-6}	53	222
128×256	TAP, $\varepsilon_{\mathrm{tr}}=10^{-4}$	9.9×10^{-15}	3.3×10^{-14}	3.0×10^{-14}	2.0×10^{-6}	51	208
128×256	DR, $\varepsilon_{\mathrm{tr}}=10^{-10}$	5.6×10^{-13}	2.2×10^{-12}	6.7×10^{-13}	2.9×10^{-13}	84	230
128×256	TAP, $\varepsilon_{\mathrm{tr}}=10^{-10}$	9.3×10^{-15}	3.4×10^{-14}	2.9×10^{-14}	1.4×10^{-12}	91	208

Table 5: Correction as a positivity limiter in the LoMaC low-rank scheme [14] for the bump-on-tail run, $T=30$, applied every 1000 steps, comparing the convex DR limiter (no rank cap) with the rank-constrained TAP limiter at rank $r=N+3$ ($N=$ current solution rank). The negativity is $\|\min\{\boldsymbol{F},\boldsymbol{0}\}\|_F\,/\,\|\boldsymbol{F}\|_F$ for the limited (post-application) solution. The LoMaC scheme conserves mass, momentum and energy to machine precision. TAP keeps $\boldsymbol{X}_k\boldsymbol{B}=\boldsymbol{0}$ exactly at every iterate, so the limited run conserves all three to machine precision. DR enforces $\boldsymbol{X}\boldsymbol{B}=\boldsymbol{0}$ only at convergence, so it preserves conservation to its orthogonality residual ($\sim 10^{-10}$ at $\varepsilon_{\mathrm{tr}}=10^{-4}$, $\sim 10^{-12}$ at $\varepsilon_{\mathrm{tr}}=10^{-10}$). Both drive the negativity to the re-truncation floor $\varepsilon_{\mathrm{tr}}$, with the kept rank rising as $\varepsilon_{\mathrm{tr}}$ is tightened. Both grids use CFL $=0.1$. The per-step cost of the LoMaC integration scales as $\mathcal{O}((m+n)r^2)$, linear in the grid size and quadratic in the rank r, so it remains a low-rank cost even as the limiter raises r. Enforcing nonnegativity increases the rank only modestly and transiently: at $\varepsilon_{\mathrm{tr}}=10^{-4}$, the scheme's own truncation level, the rank is essentially unchanged, whereas at $\varepsilon_{\mathrm{tr}}=10^{-10}$ it roughly doubles (for instance $28\to53$ on 64×128 and $49\to91$ on 128×256), but this higher rank is sustained only briefly after each application before the per-step truncation relaxes it. The time-averaged overhead is therefore only 20–30% and stays far below the full-grid $\mathcal{O}(mn\min(m,n))$ cost. CPU times are indicative and machine-dependent.

One might worry that the numerical rank of the limited solution, which reaches a sizeable fraction of $\min(N_x,N_v)$ on the 128×256 grid of Figure 7, would approach full rank as the mesh is refined. It does not: the near-full appearance is an artifact of the small $\min(N_x,N_v)$ at this resolution, not of the low-rank structure. To exhibit the trend cheaply we run the same test on the coarser 32×64 and 64×128 grids (Figure 8). The peak numerical rank grows only sub-linearly with

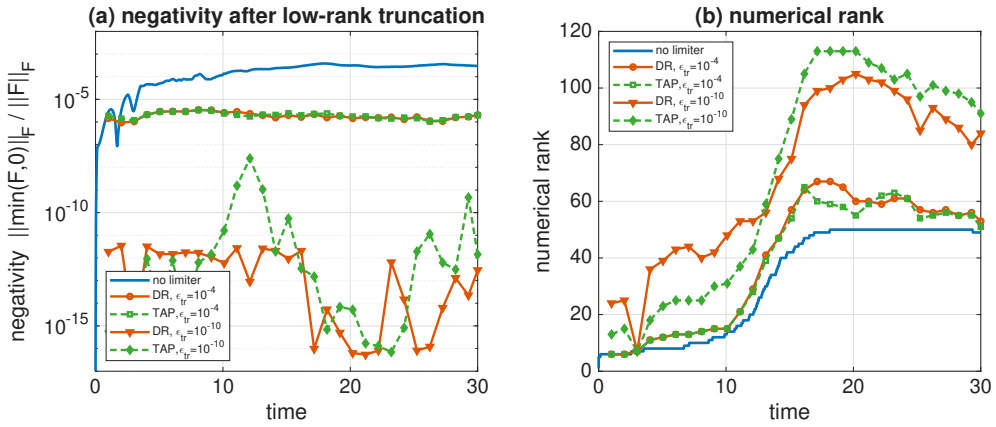


Figure 7: Positivity limiter in the LoMaC low-rank scheme for the bump-on-tail run (128×256, $T = 30$), applied every 1000 steps, comparing the convex DR limiter (no rank cap) with the rank-constrained TAP limiter at rank $r = N+3$. (a) Negativity $\|\min\{\boldsymbol{F}, \boldsymbol{0}\}\|_F / \|\boldsymbol{F}\|_F$ of the stored low-rank field versus time. The correction enforces nonnegativity to machine precision *before* truncation ($\boldsymbol{A}+\boldsymbol{X} \geqslant \boldsymbol{0}$ identically). The residual negativity plotted here is reintroduced *solely* by re-expressing the corrected field in the rank-bounded low-rank format, by discarding singular values below ε_{tr}, so it scales with ε_{tr}. Without the limiter it grows to $\sim 3 \times 10^{-4}$. With either limiter each application resets it to this re-truncation floor, $\sim 2 \times 10^{-6}$ at $\varepsilon_{\text{tr}} = 10^{-4}$ and $\sim 10^{-12}$ at $\varepsilon_{\text{tr}} = 10^{-10}$ (DR and TAP curves overlap). (b) Numerical rank: enforcing nonnegativity raises the kept rank above the no-limiter value (49), more so as ε_{tr} is tightened ($\approx$ 51–53 at $\varepsilon_{\text{tr}} = 10^{-4}$, $\approx$ 84–91 at $\varepsilon_{\text{tr}} = 10^{-10}$). The rank is reduced only by the scheme's conservative truncation, never by a mechanical cap, and DR and adaptive-rank TAP track each other.

the grid (for the no-limiter run it is 18, 28, 50 across 32×64, 64×128, 128×256, each refinement multiplying it by well under the factor 2 that a full-rank field would require), so as a fraction of the maximum $\min(N_x, N_v)$ it *decreases* under refinement. On the coarsest grid the tight-tolerance ($\varepsilon_{\mathrm{tr}} = 10^{-10}$) limiter is pinned at the full-rank ceiling, whereas on 128×256 it already sits at 80–88% of it. Doubling the resolution therefore makes the rank a smaller fraction of full, not larger, confirming that the limited solution remains genuinely low rank.

5.6 Discussion

We summarize the main observations from the experiments of this section, which comprise a fixed-size correction problem (the 64×128 Landau data, Table 1 and Figures 2–4), a cost-scaling study across four grid sizes (Section 5.4), and the use of the correction as a positivity limiter in a time-dependent solver (Section 5.5).

Approximation quality. All the algorithms produce corrections of comparable quality. For $a \geqslant 10$ in the convex formulation and for every rank constraint r tested in TAP, the relative correction $\|\boldsymbol{X}\|_F / \|\boldsymbol{A}\|_F$ is close to the theoretical lower bound (here 3.08%) and decreases monotonically as a or r increases. All convex algorithms reach the same global minimizer (3.29%, rank 33), reflecting the global optimality of the convex formulation, and TAP attains a comparable correction (3.31%) at the prescribed lower rank $r = 20$. The penalty a and the rank bound r play the same role of trading correction quality against the rank of $\boldsymbol{X}$. The naive low-rank baseline matches the rank and orthogonality constraints but cannot enforce nonnegativity ($\min(\boldsymbol{A} + \boldsymbol{X}) < 0$), and its smaller relative correction (2.93%, below the 3.08% bound) is possible only because it violates nonnegativity, which is exactly the property the optimization-based corrections are designed to deliver.

Convergence speed. On the fixed-size problem, Douglas–Rachford splitting converges in the fewest iterations among the convex algorithms (about 65 at $a = 1000$, Table 1), once its step size and relaxation are tuned (Figure 3). The dual methods are slower. Among them the gradient-based restart performs best, the L-BFGS method needs few iterations but an expensive line search, and the PR+ conjugate gradient is the slowest and does not meet the tolerance within 1000 iterations. The tangent-space accelerated alternating projection converges in the fewest iterations overall (about 45).

Per-iteration cost and scaling. The two formulations differ fundamentally in per-iteration cost: every convex iteration requires a full $m \times n$ SVD ($\mathcal{O}(mn \min(m, n))$), whereas a TAP iteration requires only a $2r \times 2r$ SVD ($\mathcal{O}(mnr)$). The scaling study confirms this empirically. The per-iteration cost ratio of DR to TAP grows from order one to about 18 as the grid is refined from 32×64 to 256×512 (Table 2 and Figure 6(a)). With its step size tuned at each grid, DR converges in fewer iterations than TAP, but its much higher per-iteration cost dominates the total wall-clock time: the two methods are comparable on the smaller grids and TAP is about $4\times$ faster on the finest grid (Table 3 and Figure 6(b)), an advantage that widens under refinement since the cost gap scales as m/r. TAP is therefore the most cost-efficient method, increasingly so at scale, while the convex formulation offers global optimality and lets the correction rank adapt freely.

Constraint preservation. The formulations also differ in which constraint the iterate satisfies before convergence. TAP keeps the orthogonality $\boldsymbol{X}_k \boldsymbol{B} = \boldsymbol{0}$ exactly at every iterate, and the iterate

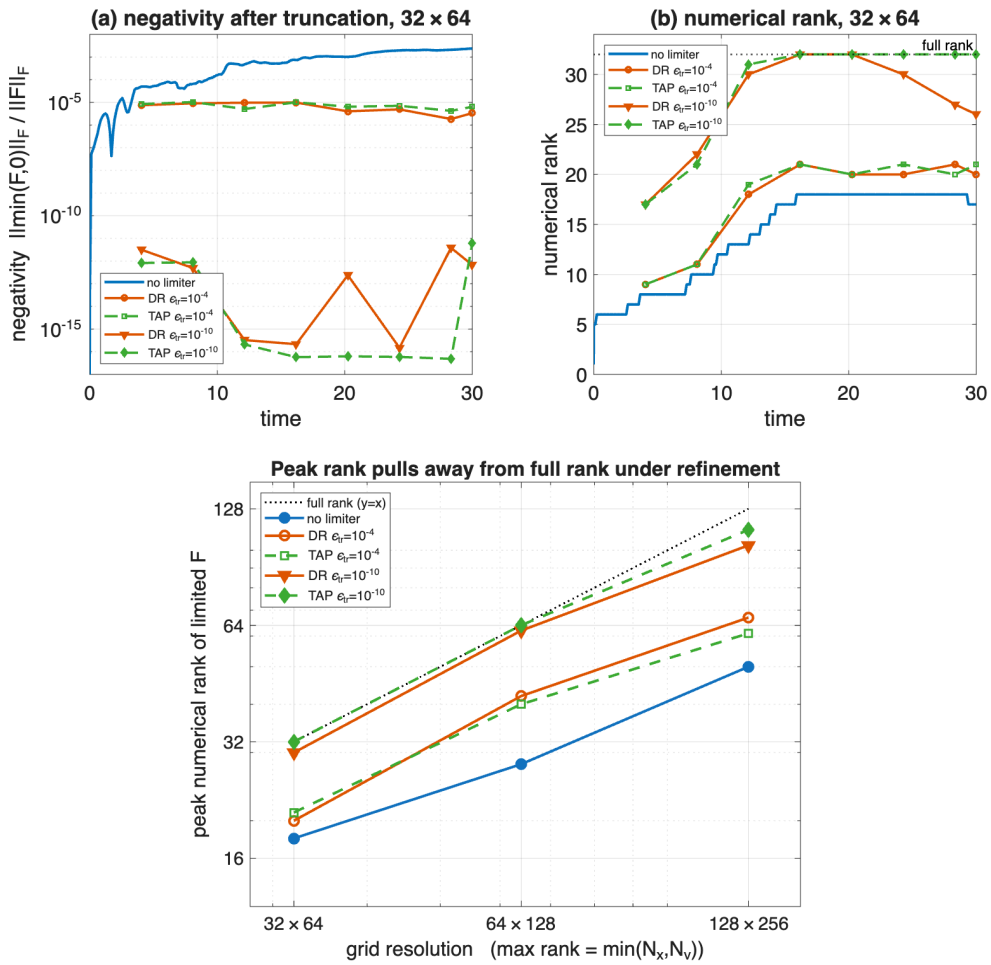


Figure 8: Rank of the limited solution versus mesh resolution, addressing whether it approaches full rank under refinement. Top: the coarse-grid (32×64) counterpart of Figure 7, showing the negativity (left) and the numerical rank (right) versus time; on this small grid the tight-tolerance limiter is pinned at the full-rank ceiling $\min(N_x, N_v) = 32$ (dotted). Bottom: peak numerical rank of the limited solution versus grid resolution for each configuration, against the full-rank line $y = \min(N_x, N_v)$ (dotted). Every curve pulls away from the full-rank line as the mesh is refined, since the absolute rank grows sub-linearly, so the near-full rank seen in Figure 7 is a coarse-grid artifact and doubling the resolution reduces the rank as a fraction of the maximum. All runs use CFL $= 0.1$.

stays low rank, whereas the convex methods that recover the primal variable by clipping (DR, dual FISTA I, dual AGD, PR+ CG, L-BFGS) keep the nonnegativity $\boldsymbol{A} + \boldsymbol{X}_k \geqslant \boldsymbol{0}$ exactly at every iterate. The dual FISTA variant II is the mirror case, keeping $\boldsymbol{X}_k\boldsymbol{B} = \boldsymbol{0}$ exactly instead. Both constraints hold at convergence, but the per-iterate property matters when the iteration is stopped early or embedded in a larger computation.

Use as a positivity limiter. The last experiment (Section 5.5) embeds the correction as a positivity limiter in the conservative LoMaC low-rank Vlasov solver. Both the convex (DR) and the rank-constrained (TAP) corrections remove the negativity introduced by the SVD truncation while leaving the conserved moments untouched, since $\boldsymbol{X}\boldsymbol{B} = \boldsymbol{0}$. The per-iterate constraint preservation above is decisive here: TAP, which keeps $\boldsymbol{X}_k\boldsymbol{B} = \boldsymbol{0}$ exactly, preserves mass, momentum and energy to machine precision, whereas DR preserves them only to its orthogonality residual. The correction itself enforces nonnegativity to machine precision, so the only residual negativity comes from re-expressing the corrected field in the rank-bounded low-rank format, and its level is set by the re-truncation tolerance, reaching machine zero as that tolerance is tightened at the cost of a higher but still moderate rank. For this to succeed, the correction rank must be allowed to track the solution rather than be fixed in advance.

6 Conclusion

We developed optimization-based post-processing algorithms to recover nonnegativity in low-rank numerical solutions of Vlasov dynamics while preserving the macroscopic conservation laws pointwise. The conservation requirement is written as an orthogonality constraint on the correction term. We studied two optimization formulations. The first is the convex problem (2) based on squared nuclear norm minimization. For this problem, Theorem 1 shows that the proximal operator can be computed through an implicit singular value thresholding equation with a threshold determined by bisection. We developed five algorithms for this formulation: Douglas–Rachford splitting, restarted dual FISTA, restarted dual accelerated gradient descent, dual PR+ conjugate gradient, and dual L-BFGS. The second is the rank-constrained non-convex formulation (17). For this formulation, we developed a tangent-space accelerated alternating projection algorithm that only requires a $2r \times 2r$ SVD per iteration. The numerical results for the Landau damping test case show that all algorithms produce comparable correction quality. The TAP algorithm is the most cost-efficient because of its low per-iteration cost, and this advantage becomes more pronounced as the problem size increases. Among the convex algorithms, Douglas–Rachford splitting converges the fastest. The gradient-restarted dual methods give the best performance among the dual approaches. The convex formulation has the advantage of global optimality, while the TAP algorithm naturally preserves low-rank storage throughout the iteration. Finally, we demonstrated that the correction can be used as a positivity limiter inside the time-dependent LoMaC low-rank scheme, which conserves mass, momentum and energy, using either the convex DR algorithm or the rank-constrained TAP algorithm with its rank tracking the solution ($r = N + 3$). On the validated, genuinely low-rank bump-on-tail run up to $T = 30$, both limiters remove the truncation negativity. Because the correction rank is not capped a priori, enforcing nonnegativity raises the kept rank, and the residual negativity is set by the re-truncation tolerance, driven toward machine precision as that tolerance is tightened. Since TAP maintains the orthogonality constraint exactly at every iterate, the TAP limiter preserves all three conserved quantities to machine precision, whereas DR

preserves them to its orthogonality residual. Both methods incur a modest, comparable overhead.

Declaration of AI-assisted technologies

All original ideas, as well as codes, are attributed to the authors. During the preparation of this work, the authors used Anthropic's Claude Code to assist with mathematical discussions, further numerical implementation, and drafting of the manuscript. After using this tool, the authors carefully reviewed and edited the content as needed and take full responsibility for the final publication.

Declarations

Funding. X. Zhang is partially supported by NSF DMS-2208518. J.-M. Qiu acknowledges support provided by Air Force Office of Scientific Research FA9550-24-1-0254 and Department of Energy DE-SC0023164. S. Becker acknowledges support from the Department of Energy DE-SC0023346.

Competing Interests. The authors have no competing interests to declare that are relevant to the content of this article.

Data and Code Availability. The test data and the code used to generate the numerical results in this paper are available from the authors upon request.

Author Contributions. All authors contributed to the conception of the study, the methodology, the numerical implementation, and the writing of the manuscript. All authors read and approved the final manuscript.

References

[1] A. Beck. *First-Order Methods in Optimization*, volume 25 of *MOS-SIAM Series on Optimization*. SIAM, Philadelphia, PA, 2017.

[2] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

[3] J.-F. Cai, E. J. Candès, and Z. Shen. A singular value thresholding algorithm for matrix completion. *SIAM Journal on optimization*, 20(4):1956–1982, 2010.

[4] G. Ceruti, J. Kusch, and C. Lubich. A rank-adaptive robust integrator for dynamical low-rank approximation. *BIT Numerical Mathematics*, 62(4):1149–1174, 2022.

[5] G. Ceruti and C. Lubich. An unconventional robust integrator for dynamical low-rank approximation. *BIT Numerical Mathematics*, 62(1):23–44, 2022.

[6] Y. Chen, D. Xiu, and X. Zhang. On enforcing non-negativity in polynomial approximations in high dimensions. *SIAM Journal on Scientific Computing*, 47(2):A866–A888, 2025.

[7] A. Dektor and L. Einkemmer. Interpolatory dynamical low-rank approximation for the 3+ 3d Boltzmann-BGK equation. *Journal of Computational Physics*, 547:114515, 2026.

[8] L. Einkemmer and I. Joseph. A mass, momentum, and energy conservative dynamical low-rank scheme for the Vlasov equation. *Journal of Computational Physics*, 443:110495, 2021.

[9] L. Einkemmer, K. Kormann, J. Kusch, R. G. McClarren, and J.-M. Qiu. A review of low-rank methods for time-dependent kinetic simulations. *Journal of Computational Physics*, 538:114191, 2025.

[10] S. A. Goreinov, I. V. Oseledets, D. V. Savostyanov, E. E. Tyrtyshnikov, and N. L. Zamarashkin. How to find a good submatrix. In V. Olshevsky and E. Tyrtyshnikov, editors, *Matrix Methods: Theory, Algorithms and Applications*, pages 247–256. World Scientific, 2010.

[11] W. Guo, J. F. Ema, and J.-M. Qiu. A local macroscopic conservative (lomac) low rank tensor method with the discontinuous Galerkin method for the Vlasov dynamics. *Communications on Applied Mathematics and Computation*, 6(1):550–575, 2024.

[12] W. Guo and J.-M. Qiu. A low rank tensor representation of linear transport and nonlinear Vlasov solutions and their associated flow maps. *Journal of Computational Physics*, 458:111089, 2022.

[13] W. Guo and J.-M. Qiu. A conservative low rank tensor method for the Vlasov dynamics. *SIAM Journal on Scientific Computing*, 46(1):A232–A263, 2024.

[14] W. Guo and J.-M. Qiu. A local macroscopic conservative (LoMaC) low rank tensor method for the Vlasov dynamics. *Journal of Scientific Computing*, 101(3):61, 2024.

[15] N. Halko, P.-G. Martinsson, and J. A. Tropp. Finding structure with randomness: probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Review*, 53(2):217–288, 2011.

[16] O. Koch and C. Lubich. Dynamical low-rank approximation. *SIAM Journal on Matrix Analysis and Applications*, 29(2):434–454, 2007.

[17] P.-L. Lions and B. Mercier. Splitting algorithms for the sum of two nonlinear operators. *SIAM Journal on Numerical Analysis*, 16(6):964–979, 1979.

[18] C. Liu, D. Milesis, C.-W. Shu, and X. Zhang. Efficient optimization-based invariant-domain-preserving limiters in solving gas dynamics equations. *Journal of Computational Physics*, 558:114839, 2026.

[19] D. C. Liu and J. Nocedal. On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45(1–3):503–528, 1989.

[20] C. Lubich and I. V. Oseledets. A projector-splitting integrator for dynamical low-rank approximation. *BIT Numerical Mathematics*, 54(1):171–188, 2014.

[21] Y. E. Nesterov. A method of solving a convex programming problem with convergence rate $o(\frac{1}{k^2})$. *Doklady Akademii Nauk SSSR*, 269(3):543–547, 1983.

[22] Y. E. Nesterov. Gradient methods for minimizing composite functions. *Mathematical Programming*, 140(1):125–161, Aug 2013.

[23] B. O'Donoghue and E. J. Candès. Adaptive restart for accelerated gradient schemes. *Foundations of Computational Mathematics*, 15(3):715–732, Jun 2015.

[24] B. Perthame and C.-W. Shu. On positivity preserving finite volume schemes for Euler equations. *Numerische Mathematik*, 73(1):119–130, 1996.

[25] M. J. D. Powell. Convergence properties of algorithms for nonlinear optimization. *SIAM Review*, 28(4):487–500, 1986.

[26] G.-J. Song, M. K. Ng, and T.-X. Jiang. Tangent space based alternating projections for nonnegative low rank matrix approximation. *IEEE Transactions on Knowledge and Data Engineering*, 35(11):11917–11934, 2023.

[27] J. Sun and J. Zhang. Global convergence of conjugate gradient methods without line search. *Annals of Operations Research*, 103(1–4):161–173, Mar 2001.

[28] X. Tang, R. Dwaraknath, and L. Ying. Variational inference and density estimation with non-negative tensor train. *arXiv preprint arXiv:2507.21519*, 2025.

[29] B. Vandereycken. Low-rank matrix completion by Riemannian optimization. *SIAM Journal on Optimization*, 23(2):1214–1236, 2013.

[30] K. Wu, X. Zhang, and C.-W. Shu. High order numerical methods preserving invariant domain for hyperbolic and related systems. *arXiv preprint arXiv:2512.09116*, 2025.

[31] E. Ye and N. F. G. Loureiro. Quantized tensor networks for solving the Vlasov–Maxwell equations. *Journal of Plasma Physics*, 90(3):805900301, 2024.

[32] X. Zhang and C.-W. Shu. On positivity-preserving high order discontinuous Galerkin schemes for compressible Euler equations on rectangular meshes. *Journal of Computational Physics*, 229(23):8918–8934, 2010.

[33] N. Zheng, D. Hayes, A. Christlieb, and J.-M. Qiu. A Semi-Lagrangian adaptive-rank (SLAR) method for linear advection and nonlinear Vlasov-Poisson system. *Journal of Computational Physics*, 532:113970, 2025.

[34] N. Zheng, W. A. Sands, D. Hayes, A. J. Christlieb, and J.-M. Qiu. A semi-Lagrangian adaptive rank (SLAR) method for high-dimensional Vlasov dynamics. *SIAM Journal on Scientific Computing*, page accepted, 2026.

[35] S. Zheng, W. Huang, B. Vandereycken, and X. Zhang. Riemannian optimization using three different metrics for Hermitian PSD fixed-rank constraints. *Computational Optimization and Applications*, 91(3):1135–1184, 2025.