Structure preserving low-rank algorithms for plasma simulations Part 2: Robust dynamical low-rank integrators, implementation, and some theory

Lukas Einkemmer
University of Innsbruck

Structure-Preserving Scientific Computing and Machine Learning Summer School and Hackathon, UW Seattle, 2025

Link to slides: `http://www.einkemmer.net/training.html`

Robust integrators

We call a method **robust** if it is not affected by small singular values.

Robust methods are constructed by finding ways to avoid the inversion of S.

- Previously regularization has been used.

Fundamental problem: X and V are not well defined for small singular values.

$$f = \sum_i X_i \sigma_i V_i.$$

but

$$K_j(t,x) = \sum_i X_i(t,x) S_{ij}(t) \qquad \text{and} \qquad L_i(t,v) = \sum_j S_{ij}(t) V_j(t)$$

are.

Projector splitting integrator

Vlasov–Poisson equation **constrained to the low-rank manifold**

$$\partial_t f = P(f)\mathrm{RHS} = P(f)\left(-v\cdot\nabla_x f + E(f)\cdot\nabla_v f\right),$$

where $P(f)$ is the orthogonal projector onto the tangent space.
We have

$$\begin{aligned} P(f)\mathrm{RHS} &= \sum_{ij}\left(\partial_t X_i S_{ij} V_j + X_i \partial_t S_{ij} V_j + X_i S_{ij}\partial_t V_j\right) \\ &= \sum_{ij}\left(\partial_t (X_i S_{ij}) V_j - X_i \partial_t S_{ij} V_j + X_i \partial_t (S_{ij} V_j)\right) \\ &= \sum_j \langle V_j, \mathrm{RHS}\rangle_v V_j - \sum_{ij} X_i \langle X_i V_j, \mathrm{RHS}\rangle_{xv} V_j + \sum_i X_i \langle X_i, \mathrm{RHS}\rangle_x, \end{aligned}$$

where we have used the DLR equations of motions

$$\partial_t S_{ij} = \langle X_i V_j, \mathrm{RHS}\rangle_{xv}, \qquad \partial_t\Big(\sum_i X_i S_{ij}\Big) = \langle V_j, \mathrm{RHS}\rangle_v \qquad \partial_t\Big(\sum_j S_{ij} V_j\Big) = \langle X_i, \mathrm{RHS}\rangle_x$$

Projector splitting integrator

We can write

$$P(f)g = P_{\overline{V}}g - P_{\overline{V}}P_{\overline{X}}g + P_{\overline{X}}g,$$

where $P_{\overline{X}}$ and $P_{\overline{V}}$ are the orthogonal projectors on $\overline{X} = \text{span}\{X_i \colon i = 1 \dots r\}$ and $\overline{V} = \text{span}\{V_j \colon j = 1 \dots r\}$.

This suggests a **splitting**.

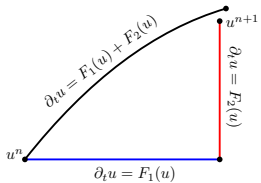
Splitting

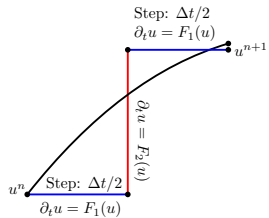
We consider

$$\partial_t u(t) = F_1(u(t)) + F_2(u(t)), \qquad u(0) = u^n,$$

where F_1 and F_2 could describe **different physics**, **different timescales**, **different coordinate axis, ...**

$$u^{n+1} = \varphi^{F_2}_{\Delta t} \circ \varphi^{F_1}_{\Delta t}(u^n) \textbf{ (Lie)} \qquad u^{n+1} = \varphi^{F_1}_{\Delta t/2} \circ \varphi^{F_2}_{\Delta t} \circ \varphi^{F_1}_{\Delta t/2}(u^n) \textbf{ (Strang)}$$





Fundamental idea of splitting is that **only subflows have to be solved**.

K step

Our goal is to solve

$$\partial_t f = P_{\overline{V}}\left(-v \cdot \nabla_x f + E(f) \cdot \nabla_v f\right).$$

We rewrite the solution using K_j as follows

$$f(t,x,v) = \sum_j K_j(t,x) V_j(t,v), \qquad \text{with} \quad K_j(t,x) = \sum_i X_i(t,x) S_{ij}(t).$$

This yields

$$\sum_j \partial_t K_j(t,x) V_j(t,v) + \sum_j K_j(t,x) \partial_t V_j(t,v)$$
$$= \sum_j \left\langle V_j(t,\cdot), v \mapsto -v \cdot \nabla_x f(t,x,v) + E(f)(t,x) \cdot \nabla_v f(t,x,v) \right\rangle_v V_j(t,v).$$

K step

The solution is given by $V_j(t,v) = V_j(0,v)$ and

$$\begin{aligned}\partial_t K_j(t,x) &= \left\langle V_j, v \mapsto v \cdot \nabla_x f(t,x,v) + E(f)(t,x) \cdot \nabla_v f(t,x,v) \right\rangle_v \\ &= -\sum_l \langle V_j v V_l \rangle_v \cdot \nabla_x K_l(t,x) + \sum_l E(f)(t,x) \cdot \langle V_j \nabla_v V_l \rangle_v K_l(t,x)\end{aligned}$$

For the **first subflow of the projector splitting algorithm** we thus obtain

$$\partial_t K_j(t,x) = -\sum_l c_{jl}^1 \cdot \nabla_x K_l(t,x) + \sum_l c_{jl}^2 \cdot E(K)(t,x) K_l(t,x),$$

The coefficients are determined as follows ($V = V^n$)

$$c_{jl}^1 = \int_{\Omega_v} v V_j^n V_l^n \,\mathrm{d}v, \quad c_{jl}^2 = \int_{\Omega_v} V_j^n (\nabla_v V_l^n) \,\mathrm{d}v.$$

Do not neglect the cost of computing the coefficients.

K step

The equation is formulated with K and V (neither X nor S are explicitly involved).

To proceed with the next step in the algorithm we have to obtain X and S.

- **Why is this approach then advantageous?**

The X and S are recovered from K by a **QR decomposition** as

$$K_j(\Delta t) = \sum_i X_i^{n+1} S_{ij}$$

Well defined even for singular $K = [K_1, \ldots, K_r]$ and gives automatically the (almost correct) orthogonality relation for the X_i.

- Result is a robust approximation even if the rank r is chosen *too large*.

Note that S is not necessarily diagonal.

S step

Our goal is to solve

$$\partial_t f = -P_{\overline{V}} P_{\overline{X}} \left(-v \cdot \nabla_x f + E(f) \cdot \nabla_v f\right).$$

The solution is $X_i(t,x) = X_i(0,x)$, $V_j(t,v) = V_j(0,v)$, and

$$\begin{aligned}\partial_t S_{ij} &= \left\langle X_i^{n+1} V_j^n, (x,v) \mapsto (v \cdot \nabla_x - E(S)(t,x) \cdot \nabla_v) \sum_{kl} X_k^{n+1}(x) S_{kl}(t) V_l^n(v) \right\rangle_{xv} \\ &= \sum_{kl} \left(c_{jl}^1 \cdot d_{ik}^2 - c_{jl}^2 \cdot d_{ik}^1[E(S(t))] \right) S_{kl}(t)\end{aligned}$$

with

$$d_{ik}^1[E] = \int_{\Omega_x} X_i^{n+1} E X_k^{n+1} \,\mathrm{d}x, \qquad d_{ik}^2 = \int_{\Omega_x} X_i^{n+1} (\nabla_x X_k^{n+1}) \,\mathrm{d}x.$$

The S step integrates backward in time.

L step

Our goal is to solve

$$\partial_t f = P_{\overline{X}}\left(-v \cdot \nabla_x f + E(f) \cdot \nabla_v f\right).$$

We define

$$f(t,x,v) = \sum_i X_i(t,x) L_i(t,v), \qquad \text{with} \quad L_i(t,v) = \sum_j S_{ij}(t) V_j(t,v).$$

The solution is $X_i(t,x) = X_i(0,x)$ and

$$\begin{aligned}\partial_t L_i(t,v) &= \left\langle X_j^{n+1}, x \mapsto (-v \cdot \nabla_x + E(L)(t,x) \cdot \nabla_v) \sum_k X_k^{n+1} L_k(t,v) \right\rangle_x \\ &= \sum_k d_{ik}^1[E(L(t,\cdot))] \cdot \nabla_v L_k(t,v) - \sum_k (d_{ik}^2 \cdot v) L_k(t,v).\end{aligned}$$

Then S and V are recovered from L by a **QR decomposition**.

Electric field

The electric field E is computed from

$$-\Delta\phi = 1 - \sum_{ij} X_i(x) S_{ij}(t) \int V_j \, \mathrm{d}v, \qquad E = -\nabla\phi.$$

In practice we usually approximate E by E^n (first order) or $E^{n+1/2}$ (second order).

- $E^{n+1/2}$ has to be approximated (to first order) in an actual implementation.

Lie projector splitting algorithm

1. Solve $-\Delta\phi^n = 1 - \rho^n$ for $\rho^n = \sum_{ij} X_i^n S_{ij}^n \int V_j^n \,\mathrm{d}v$ and $E^n = -\nabla_x \phi$.
2. Compute $c_{jl}^1 = \int_{\Omega_v} v V_j^n V_l^n \,\mathrm{d}v$ and $c_{jl}^2 = \int_{\Omega_v} V_j^n (\nabla_v V_l^n) \,\mathrm{d}v$.
3. Solve $\partial_t K_j = -\sum_l c_{jl}^1 \cdot \nabla_x K_l + \sum_l c_{jl}^2 \cdot E^n K_l$ with initial value $\sum_i X_i^n S_{ij}^n$ up to time Δt to obtain $K_j(\Delta t)$.
4. Perform a QR decomposition of $K_j(\Delta t)$ to obtain $\mathbf{X_i^{n+1}}$ and $S_{ij}^\star$.
5. Compute $d_{ik}^1 = \int_{\Omega_x} X_i^{n+1} E^n X_k^{n+1} \,\mathrm{d}x$ and $d_{ik}^2 = \int_{\Omega_x} X_i^{n+1} (\nabla_x X_k^{n+1}) \,\mathrm{d}x$.
6. Solve $\partial_t S_{ij} = \sum_{kl} \left(c_{jl}^1 \cdot d_{ik}^2 - c_{jl}^2 \cdot d_{ik}^1 \right) S_{kl}$ with initial value $S_{ij}^\star$ up to time Δt to obtain $S_{ij}^{\star\star}$.
7. Solve $\partial_t L_i = \sum_k d_{ik}^1 \cdot \nabla_v L_k - \sum_k (d_{ik}^2 \cdot v) L_k$ equation with initial value $\sum_j S_{ij}^{\star\star} V_j^n$ up to time Δt to obtain $L_i(\Delta t)$.
8. Perform a QR decomposition of $L_i(\Delta t)$ to obtain $\mathbf{V_j^{n+1}}$ and $\mathbf{S_{ij}^{n+1}}$.

Strang projector splitting

We can raise this to second (or higher) order by **Strang splitting**.

- Be careful how you treat the electric field.

The following algorithm is second order and requires only two Poisson solves.

1. Perform a **Lie projector splitting** with step size $\Delta t/2$.
2. Use the result to compute $E^{n+1/2}$.
3. **K step** with step size $\Delta t/2$ and $E^{n+1/2}$.
4. **S step** with step size Δt and $E^{n+1/2}$.
5. **L step** with step size $\Delta t/2$ and $E^{n+1/2}$.

Why does it work and a bit of theory

Theory

For the **Lie projector splitting integrator** we assume

- F is Lipschitz continuous;
- At each time solution is approximated by rank r with an error of at most ϵ.

Theorem

The error satisfies for all n with $t^n = n\Delta t \leq T$

$$\|f_r^n - f(t^n)\| \leq c_1\epsilon + c_2\Delta t,$$

where f_r^n *is the low-rank solution at time* t^n.

Not a PDE result.

- Time and space discretization error and, for us, approximation of the electric field.

Exactness property

Under certain assumptions the projector splitting integrator is exact.

Theorem

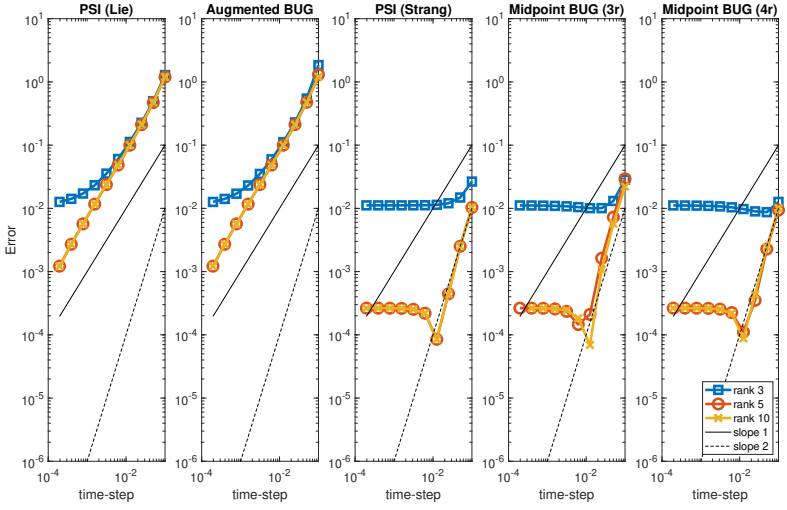
Assume that the solution is rank $\hat{r}$ for all times and $r \geq \hat{r}$. Then

$$\|f_r^n - f(t^n)\| = 0.$$

That is, $c_2 \to 0$ as $\epsilon \to 0$.

Not very relevant for realistic problems.

Typical convergence behavior



G. Ceruti, L. Einkemmer, J. Kusch, C. Lubich. BIT Numer. Math. 64:30, 2024.

Is the solution low-rank

A **low-rank method can only work well if the solution is low-rank.**

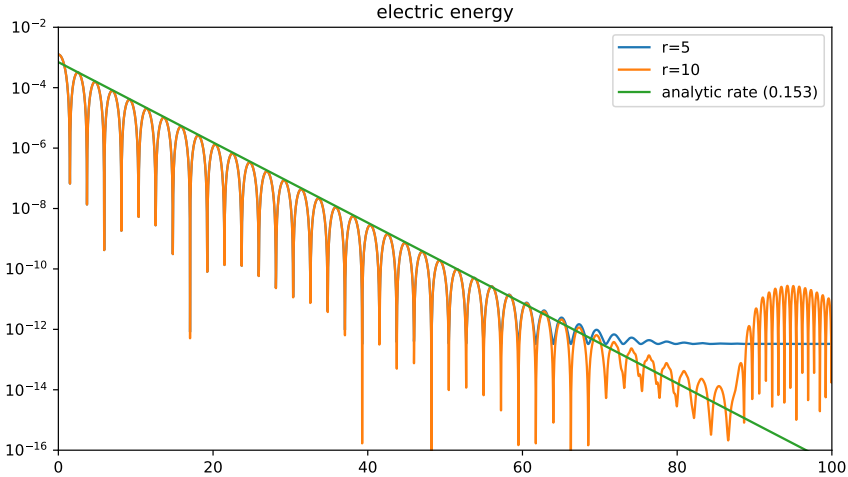
- I.e., ϵ is small for $r \ll n^d$.

This is problem specific and conveniently assumed to hold in the theory.

Is it true for kinetic equations?

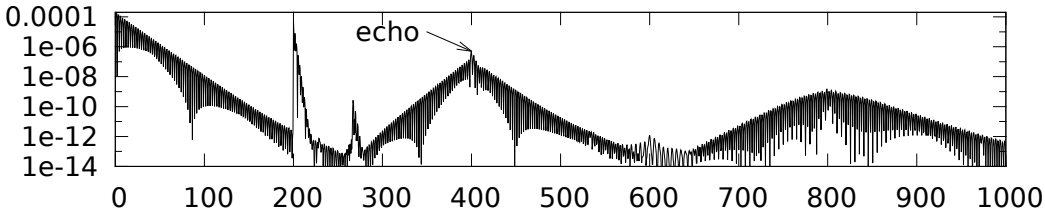
Linear Landau damping

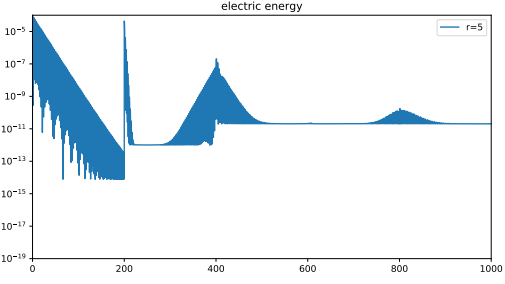
Low-rank approximation with 256 grid points in each direction.

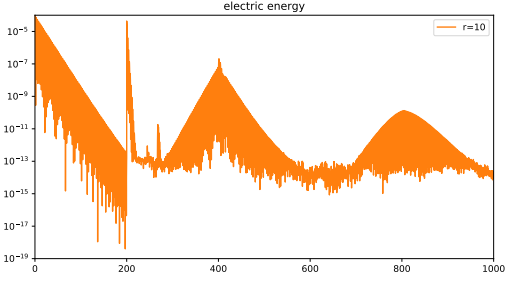


Plasma echo

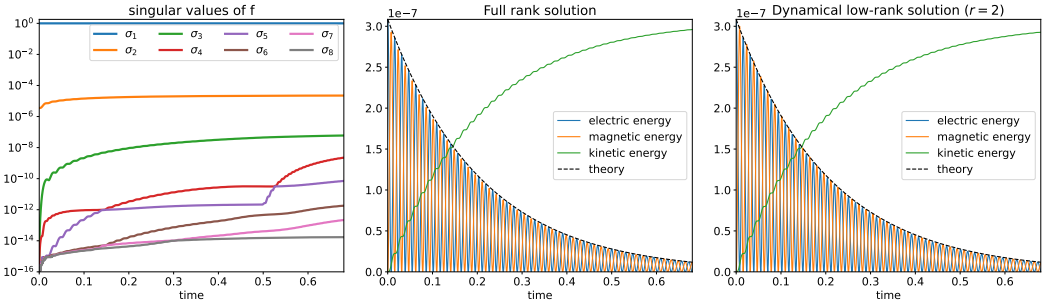
Plasma echo with 512×4096 grid points.







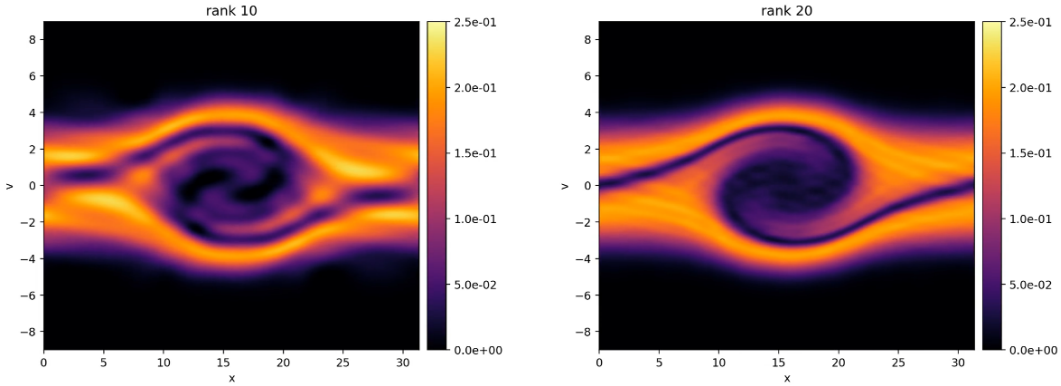
Alfvén waves



Observation 1: Low-rank methods are extremely effective for (linear) wave propagation.

Two-stream instability

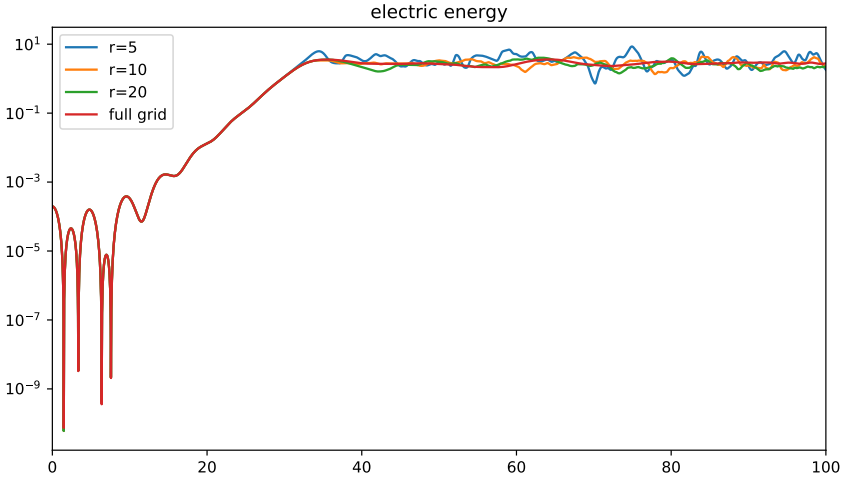
Low-rank approximation with 512 grid points per direction ($r = 10$ left, $r = 20$ right).



[Video]

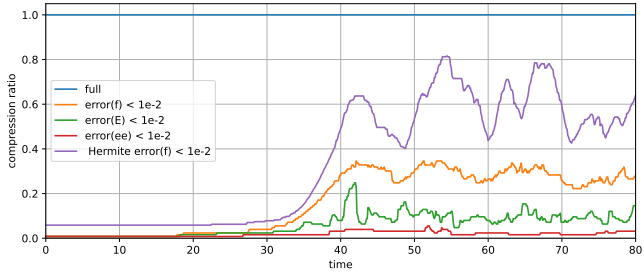
Two-stream instability

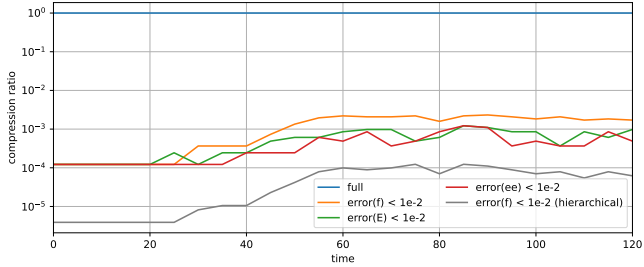
Time evolution of the electric energy.



Compression for a bump-on-tail instability

Observation 2: In many situations low-rank can give good results even in the nonlinear regime.





Linear regime

Why does low-rank work so well in the linear regime?

We consider a small perturbation around the equilibrium $f^{(0)}(v)$

$$f(t,x,v) = f^{(0)}(v) + f^{(1)}(t,x,v), \qquad E(t,x) = 0 + E^{(1)}(t,x).$$

This results in the **linearized Vlasov equation**

$$\partial_t f^{(1)}(t,x,v) + v \cdot \nabla_x f^{(1)}(t,x,v) + E^{(1)}(x) \cdot \nabla_v f^{(0)}(v) = 0,$$

where we have dropped the second order term $E^{(1)}(x) \cdot \nabla_v f^{(1)}(v)$.

Fourier transform (in x) the **Vlasov–Poisson** equation

$$\partial_t \hat{f}_k^{(1)}(t,v) + iv \cdot k \hat{f}_k^{(1)}(t,v) + \hat{E}_k^{(1)} \cdot \nabla_v f^{(0)}(v) = 0,$$
$$\hat{E}_k^{(1)} = -\frac{k}{k^2} \int \hat{f}_k^{(1)}(t,v)\,\mathrm{d}v, \qquad k \neq 0.$$

Linear regime

Now let us assume

$$f(0,x,v) = f^{(0)}(v) + \sum_{i=1}^{m} \hat{f}^{(1)}_{k_i}(0,v)\mathrm{e}^{ik_i x}.$$

E.g. Landau damping with $m = 2$ (rank 1).

Since the linear problem does not excite any new Fourier modes

$$f(t,x,v) = f^{(0)}(v) + \sum_{i=1}^{m} \hat{f}^{(1)}_{k_i}(t,v)\mathrm{e}^{ik_i x}$$

which is at most rank $m+1$.

Our low-rank algorithm is **more general than the previous analysis suggests** (i.e. in general $X_i(t,x) \neq \mathrm{e}^{ikx}$).

- The low-rank algorithm **captures saturation** perfectly.

L.E., A. Ostermann, C. Piazzola, J. Comput. Phys. 403, 2020.

Smoothness & adaption in velocity

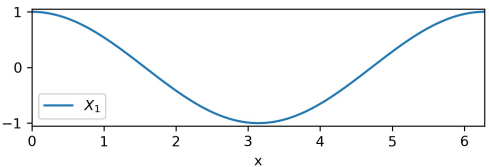
The low-rank algorithm is able to **resolve filamentation**. Consider

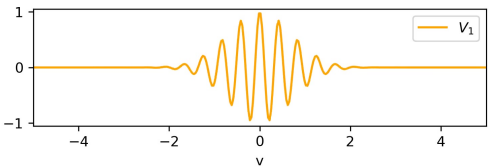
$$\partial_t f(t,x,v) + v \cdot \nabla_x f(t,x,v) = 0, \qquad f(0,x,v) = e^{ikx} e^{-v^2}.$$

The solution

$$f(t,x,v) = e^{ik(x-vt)} e^{-v^2} = e^{ikx} e^{-ikvt} e^{-v^2}.$$

is still **rank 1**.





[Video]

Smoothness in v is not necessary for low-rank approximations.

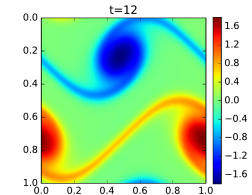
- Hermite fails here because it require a large number of basis functions to resolve v.

Limits that explicitly have a low-rank structure

Collisional kinetic equations have a diffusive or fluid limit

$$\partial_t f(t,x,v) + \frac{1}{\epsilon^\alpha} v \cdot \nabla_x f(t,x,v) = \frac{1}{\epsilon^\beta}(f_{\text{eq}}(f) - f) \xrightarrow{\epsilon \to 0}$$

Euler/heat equation



Diffusion limit is rank 1.

In the **incompressible fluid limit we are approximately low-rank**

$$\frac{\rho}{(2\pi)^{d/2}} \exp\left(-\tfrac{1}{2}(v-u)^2\right) \approx \frac{\rho}{(2\pi)^{d/2}} \exp\left(-\frac{v^2}{2}\right)\left(1 + v \cdot u + \frac{(v\cdot u)^2}{2} - \frac{u^2}{2}\right) + \mathcal{O}\left(u^3\right).$$

Rank 6 in 2D and rank 10 in 3D (compare to 27 directions often used in LB methods).

- Simulation for $n_x = 128$, $n_v = 16$, and $\text{Re} = 1000$.

L.E. SIAM. J. Sci. Comput. 41, 2019.
Z. Ding, L.E., Q. Li. SIAM J. Numer. Anal. 59:4, 2021.
L.E., J. Hu, Y. Wang. J. Comput. Phys. 439, 2021.

Implementation

Implementation

Discretized system

$$f = XSV^T$$

with

$$f_{kl} = f(t, x_k, v_l), \qquad X_{ki} = X_i(t, x_k), \qquad V_{lj} = V_j(t, v_l).$$

In **matrix form**

$$X(t) = \underbrace{\begin{bmatrix} X_1(t,x_1) & \cdots & X_r(t,x_1) \\ \vdots & \ddots & \vdots \\ X_1(t,x_n) & \cdots & X_r(t,x_n) \end{bmatrix}}_{\in \mathbb{R}^{n_x \times r}}, \qquad V(t) = \underbrace{\begin{bmatrix} V_1(t,v_1) & \cdots & V_r(t,v_1) \\ \vdots & \ddots & \vdots \\ V_1(t,v_m) & \cdots & V_r(t,v_m) \end{bmatrix}}_{\in \mathbb{R}^{n_v \times r}}.$$

with $S \in \mathbb{R}^{r \times r}$. **Never explicitly form f!**

Space discretization

K step (one-dimensional case)

$$\partial_t K_j = -\sum_l c_{jl}^1 \cdot \nabla_x K_l + \sum_l c_{jl}^2 \cdot E^n K_l$$

becomes

$$\partial_t K = -A_{\partial_x} K (c^1)^T + \mathrm{diag}(E^n) K (c^2)^T,$$

where A_{∂_x} is the discretization of the spatial derivative (e.g. second order central differences).

Compute the coefficients by quadrature, e.g.

$$c^1 = \Delta v V^{n,T} \mathrm{diag}(\{v_l\}_l) V^n \approx \int v V_j^n V_l^n \,\mathrm{d}v, \qquad c^2 = \Delta v V^{n,T} A_{\partial_v} V^n \approx \int V_j^n (\nabla_v V_l^n) \,\mathrm{d}v.$$

Time discretization

Coefficients c^1 and c^2 do not change during the K step.

- Needs to be computed only once!

Add appropriate time discretization (e.g. RK4) to

$$\partial_t K = -A_{\partial_x} K (c^1)^T + \mathrm{diag}(E^n) K (c^2)^T,$$

where K is the only unknown.

Many other choices are possible

- FFT based spectral methods combined with splitting
- Semi-Lagrangian schemes
- Runge–Kutta discontinuous Galerkin schemes
- Runge–Kutta with a high-order upwind scheme

L^2 vs ℓ^2

We consider discretization of functions space where L^2 is the natural norm. Thus,

$$\langle f(x), g(x) \rangle_{L^2} = \int f(x) g(x)\, dx \approx \Delta x \sum_m f_m g_m.$$

This is not the same (by the constant factor Δx) as

$$\langle f, g \rangle_{\ell^2} = \sum_m f_m g_m.$$

In this case, $\langle f, g \rangle_{\ell^2} \to \infty$ for $\Delta x \to 0$.

In particular, for the QR decomposition

```
X, S = qr(K, mode='reduced')
X *= 1.0/sqrt(hx)
S *= sqrt(hx)
```

Literature

Literature

[C. Lubich, I.V. Oseledets. BIT Numer. Math. 54(1) 2014]

- The original work on the projector splitting integrator (the first robust integrator).
- Rigorous analysis.

[L.E., C. Lubich, SIAM J. Sci. Comput. 40(5), 2018]

- Projector splitting based dynamical low-rank algorithm for Vlasov–Poisson.
- The original work for dynamical low-rank and kinetic equations.

[L.E., A. Ostermann, C. Piazzola. J Comput. Phys. 403, 2020]

- Analysis of the linear regime.
- Projector splitting integrator for Vlasov–Maxwell with divergence correction.

[L.E, J. Hu, Y. Wang. J. Comput. Phys. 439, 2021]

- A good starting point for dynamical low-rank in the collision dominated regime/asymptotic preserving schemes.