Structure preserving low-rank algorithms for plasma simulations
Part 1: Introduction to dynamical low-rank algorithms

Lukas Einkemmer
University of Innsbruck

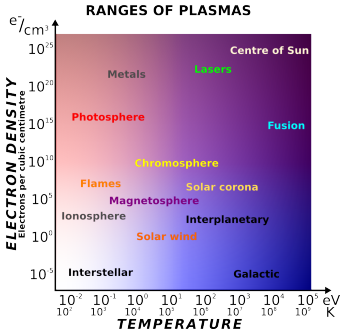
Structure-Preserving Scientific Computing and Machine Learning Summer School and Hackathon, UW Seattle, 2025

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

What is a plasma

Plasma is a state of matter where **electrons and ions are not bound to each other**.

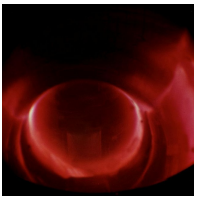
- A gas heated to sufficiently high temperature.
- An ionized gas rarefied enough such that recombination is slow.
- Electrons moving freely in a metal.



Plasma is the most common state of matter in the universe.

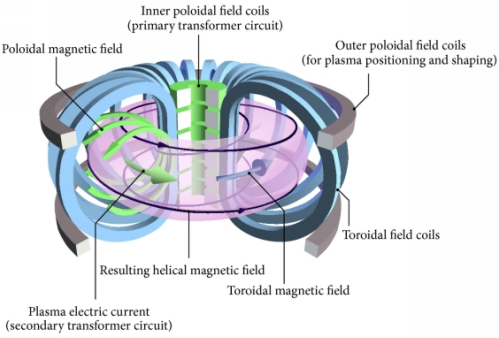






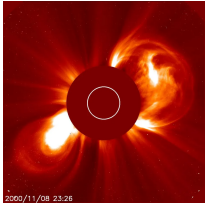
Fusion

In a **tokamak** fusion reactor magnetic fields compress and stabilize the plasma.



Promise of emission free and cheap energy from hydrogen.

Only one fusion reactor with $Q > 1$.



Plasma systems are inherently unstable!

Images from `https://dx.doi.org/10.1155/2014/940965` and `https://tinyurl.com/cmesunobs`.

Newton's law

Newton's law: $F = ma$ with acceleration $a = \ddot{x}$, mass m, and force F.

Multi-particle system:

$$\dot{x}_i(t) = v_i(t), \qquad \dot{v}_i(t) = F(x_i(t))/m_i, \qquad x(t) \in \mathbb{R}^N,\ v(t) \in \mathbb{R}^N.$$

The ith particle is described by position x_i, velocity v_i, and mass m_i.

For practical systems N is extremely large (order of **Avogadro constant** $\approx \mathbf{10^{23}}$).

- Even if we could track the position of each particle this is usually not interesting.
- Macroscopic quantities (density, momentum density, ...) much more important.

Kinetic description

We introduce a **particle-density** $f(t, x, v)$ such that

$$\int_{x_1}^{x_2} \int_{v_1}^{v_2} f(t, x, v)\, \mathrm{d}(x, v) = \text{number of particles with } x \in [x_1, x_2] \text{ and } v \in [v_1, v_2].$$

Number of particles is conserved

$$\partial_t f(t, x, v) + \nabla_{x,v} \cdot \left(f(t, x, v) \begin{bmatrix} v \\ a \end{bmatrix} \right) = 0.$$

Acceleration $a = F/m$ determined by Newton's law. Yields **kinetic equation**

$$\partial_t f(t, x, v) + v \cdot \nabla_x f(t, x, v) + \frac{F}{m} \cdot \nabla_v f(t, x, v) = 0.$$

Often F self-consistently couples to f (i.e. F depends on f).

Kinetic description

The force field F is not very useful to model collisions. **Boltzmann equation:**

$$\partial_t f + v \cdot \nabla_x f + \frac{F}{m} \cdot \nabla_v f = C(f).$$

First order hyperbolic equation with

Transport:	$f(t,x,v) = f(0, x - vt, v)$
Acceleration:	$f(t,x,v) = f(0, x, v - tF/m)$
Collision:	usually only acts in v.

For **collisionless problems** (i.e. $C(f) = 0$) particles travel along the **characteristics given by Newton's law**.

Fluid models, such as the Euler equation and Navier–Stokes equation, can be derived by assuming $f(t,x,v) \propto \rho \exp(-(v-u)^2/(2T))$, i.e. thermodynamic equilibrium.

Vlasov–Poisson equation

Interest from applications such as Tokamak devices (fusion energy), astrophysical plasmas (space weather, magnetosphere, star formation), radiative transfer, ion thrusters, laser plasma interaction, etc.

- Many large scale codes: GYSELA5D, Vlasiator, ...

In a plasma electromagnetic effects are important: $F = qE/m$.

Vlasov equation in dimensionless form

$$\partial_t f + v \cdot \nabla_x f - E \cdot \nabla_v f = 0$$

coupled to a **Poisson problem** (Gauss's law)

$$E = -\nabla\phi \qquad \text{with} \qquad -\Delta\phi = 1 - \int f \, dv.$$

Vlasov–Maxwell equations include magnetic effects.

Landau damping

Initial value

$$f(0,x,v) = (1+\alpha\cos(kx))\frac{e^{-v^2/2}}{(2\pi)^{d_v/2}}, \qquad d_v \text{ number of dimensions in the } v\text{-direction.}$$

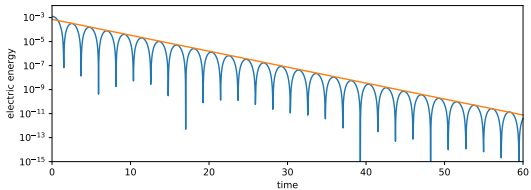
For $\alpha = 0$ we have an equilibrium.

Quantity of interest is the **electric energy** given by $\frac{1}{2}\int E^2\,\mathrm{d}x$.

- Exponential decay is **not** expected for a **hyperbolic problem**.

First described by **Landau in 1946** using linearization.

Fields Medal 2010 (Cédric Villani) for proving Landau damping and convergence to equilibrium (for α small).



L. Landau. J. Phys. (USSR) 10 (1946).
C. Mouhot, C. Villani. Acta Math. 207:1 (2011).

Bump-on-tail instability

Many fusion devices use beam heating (e.g. NBI – neutral beam injection).

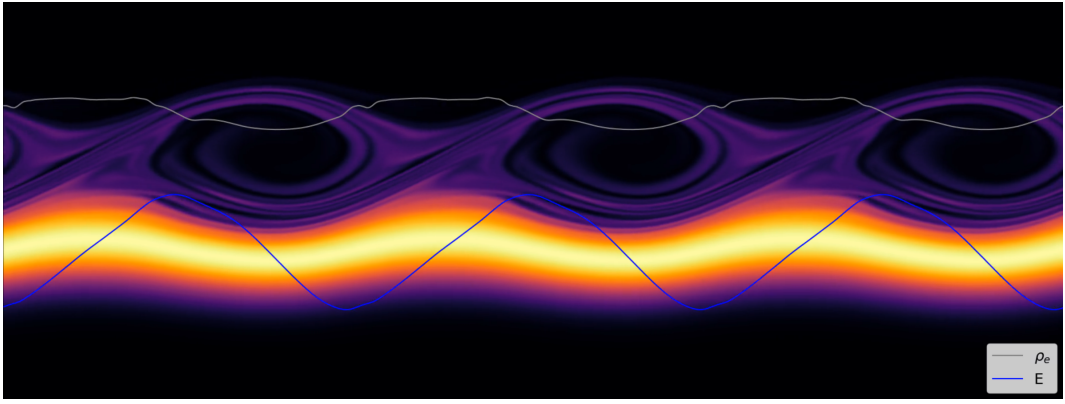
This can lead to a kinetic instability.

- System is far away from thermodynamic equilibrium.

Bump-on-tail instability

To study kinetic dynamics (in almost all situations) requires numerical simulation.

Bump-on-tail instability: $f(0, x, v) \propto \mathrm{e}^{-v^2/2} + \alpha \mathrm{e}^{-(v-v_0)^2/2}$.



[Video]

Complexity reduction, or why is this a hard problem?

Option 1: direct discretization

Solve the problem by a well optimized method (say splitting+semi-Lagrangian dG).

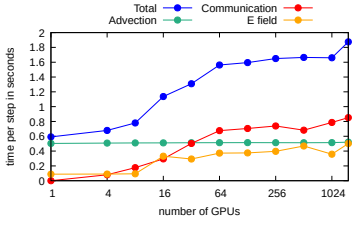
Numerical challenges:

- Six-dimensional phase space ($n = 50$: 250 GB memory vs $n = 200$: 1024 TB)
- Small scale structures force a sufficiently fine space discretization.

To obtain results for **five or six-dimensional** problems requires the **largest supercomputers currently available** (perhaps more than that).

Simulation with 1500 GPUs and $72^3 144^3$ grid points on JUWELS Booster:

- 2×24 AMD EPYC 7402 cores and $4\times$ **NVIDIA A100 GPUs per node**.
- Total of 150 TB of GPU memory.



L.E., A. Moriggl. Int. J. High Perform. Comput. Appl. 37:2 (2022).

Option 2: Particle methods

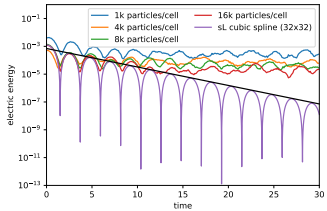
If everything else fails, **try Monte-Carlo**.

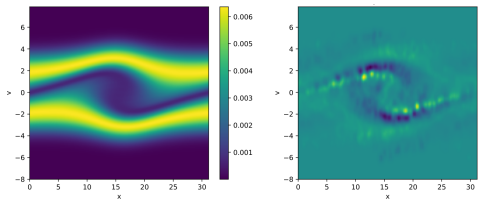
Probably still the most widely used method (especially in physics).

Particle methods have been employed extensively.

- Only x is discretized.
- Particles push and field solves are alternated.

But, suffers from **slow convergence**, numerical noise and failure to resolve the tail of the distribution.





Complexity reduction

A class of techniques to **reduce the degrees of freedom** (i.e. complexity) required to treat a problem.

Data based

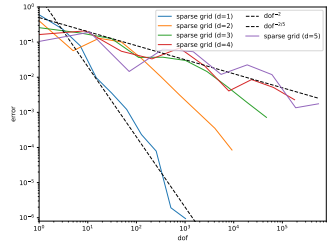
- Start with data to learn a (not too complex) model from.
- But **requires a lot of data**, **unclear how well it generalizes**, often **not interpretable**.
- Examples: Machine learning, POD type methods, etc.

Model based

- Starts with a known model (e.g. Vlasov–Poisson equation).
- Apply a series of mathematical/numerical operations to reduce the problem.

Option 3: Sparse grids

Sparse grids
[H. Bungartz, M. Griebel. Acta Numer. 13, 2004]



Consider

$$u\colon [0,1]^d \to \mathbb{R},\ x \mapsto \prod_{j=1}^{d} \exp(-\alpha(x_j - 1/2)^2)$$

for which

$$\|\partial_{x_1} \dots \partial_{x_d} u\|_\infty \approx (2\alpha)^d.$$

Why is complexity reduction hard for hyperbolic problems

Advection equation

$$\partial_t u(t,x) + v\partial_x u(t,x) = 0, \qquad \Omega = [0,1], \qquad u(0,x) = u^0(x).$$

Consider the approximation

$$u(t,x) = \sum_{i=1}^{n} u_i(t)\phi_i(x).$$

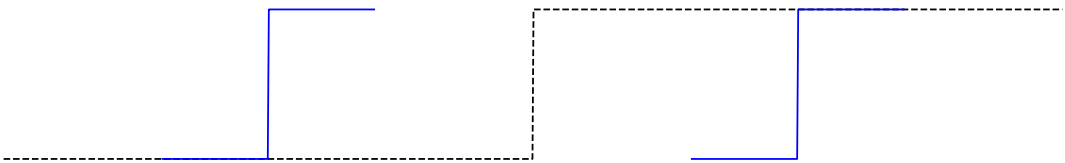
What is the best basis $\{\phi_i\}_{i=1}^n$?
Solution lives in

$$U = \{u^0(x-\alpha)\colon \alpha \in [0,1]\}$$

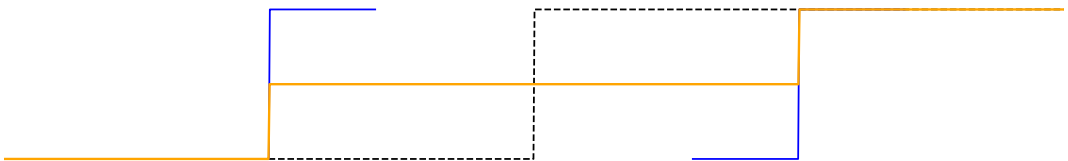
Kolmogorov N-width gives the best error for a basis with n elements

$$d_n(U) = \inf_{\dim(V_n)=n} \sup_{u\in U} \inf_{v_n\in V_n} \|u - v_n\|_{L^2(\Omega)}.$$

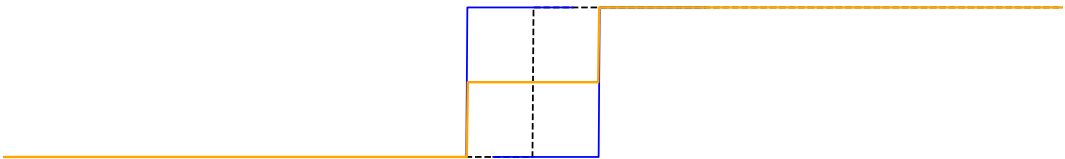
Kolmogorov N-width



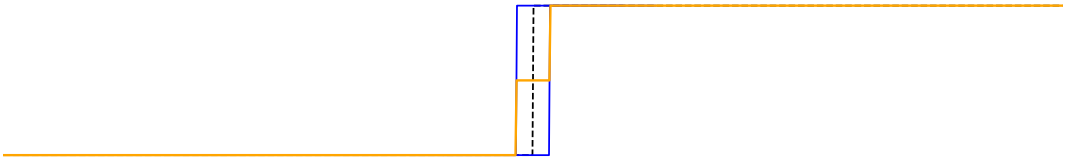
Kolmogorov N-width



Kolmogorov N-width



Kolmogorov N-width



Behavior of the error

- L^∞ error is always $\mathcal{O}(1)$
- L^2 error is proportional to n^{-1}

Kolmogorov N-width

For smooth u^0 we have $d_n(U) \leq C \exp(-cn)$ for some constant c.

- Proof: Use that Fourier coefficients of a smooth function decay exponentially.

For non-smooth u^0 we have only $d_n(U) \leq Cn^{-1/2}$.

- **Thus, any fixed basis is very inefficient.**

But if we vary the basis in time $n = 1$ is sufficient as

$$u(t,x) = 1 \cdot \phi(t,x), \qquad \text{with} \qquad \phi(t,x) = u^0(t - vt).$$

Dynamical low-rank approximation

Singular value decomposition

Singular value decomposition for a matrix $A_{ij} = g(x_i, v_j)$ is given by

$$A = VSW^T \in \mathbb{R}^{n \times m}$$

with $V \in \mathbb{R}^{n \times r}$, $S \in \mathbb{R}^{r \times r}$, $W \in \mathbb{R}^{m \times r}$, and r the rank of A.

Low-rank approximation

$$g(x, v) \approx \sum_{ij} X_i(x) S_{ij} V_j(v).$$

Orthogonality constraints: $\langle X_i, X_j \rangle_x = \delta_{ij}$, $\langle V_i, V_j \rangle_v = \delta_{ij}$.

Dynamical low-rank approximation

Step 1: We start with a (usually) high-dimensional PDE

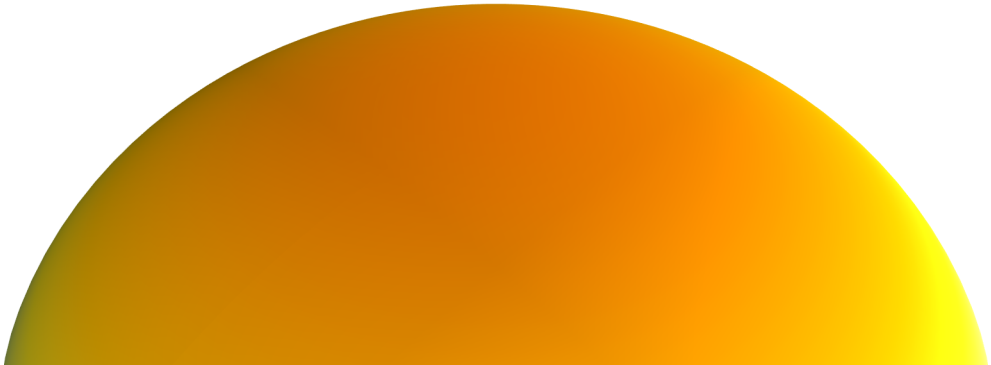
$$\text{Find } f(t,\xi) \qquad \text{such that} \qquad \partial_t f = \text{RHS}(f).$$

Dynamical low-rank approximation

Step 2: Low-rank approximation with $\xi = (x, v)$

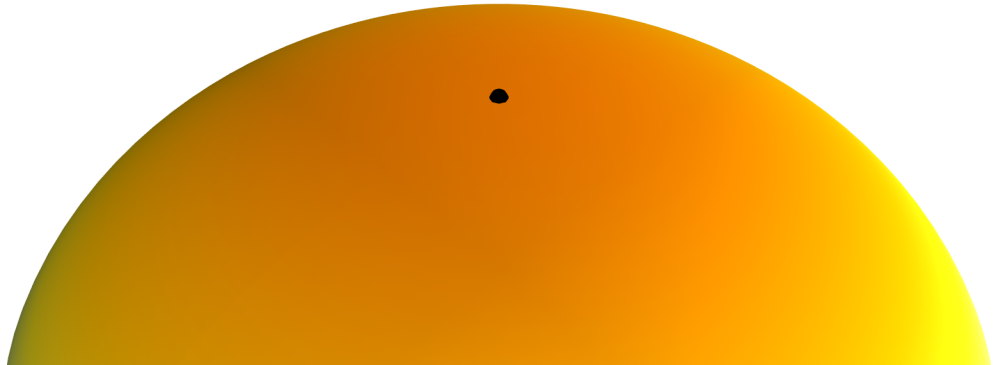
$$f(t, x, v) \approx \sum_{i=1}^{r} \sum_{j=1}^{r} X_i(t, x) S_{ij}(t) V_j(t, v), \qquad \langle X_i, X_j \rangle_x = \delta_{ij}, \quad \langle V_i, V_j \rangle_v = \delta_{ij}.$$

forms a **manifold** $\mathcal{M}_r$.



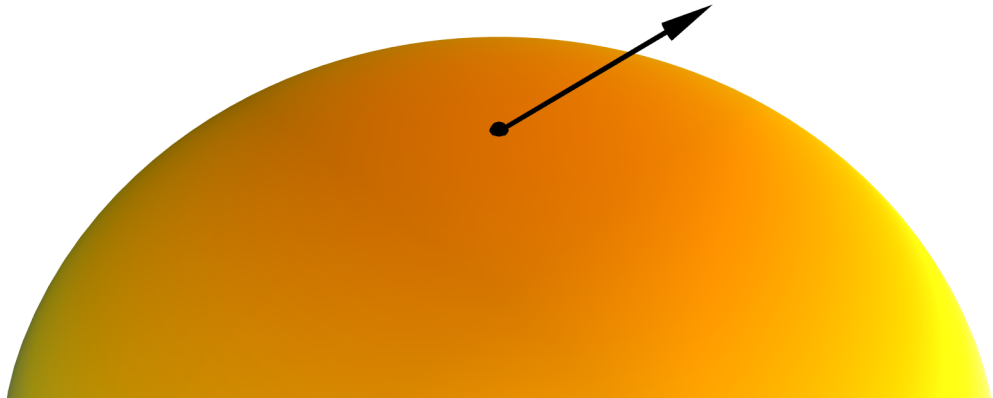
Dynamical low-rank approximation

Step 3: Exact flow $\partial_t f = \mathrm{RHS}(f)$ takes us out of the low-rank manifold.



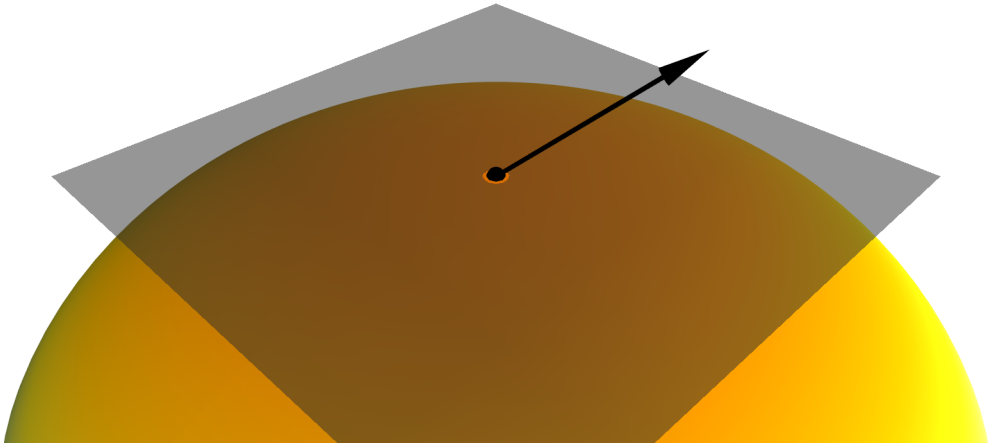
Dynamical low-rank approximation

Step 3: Exact flow $\partial_t f = \mathrm{RHS}(f)$ takes us out of the low-rank manifold.



Dynamical low-rank approximation

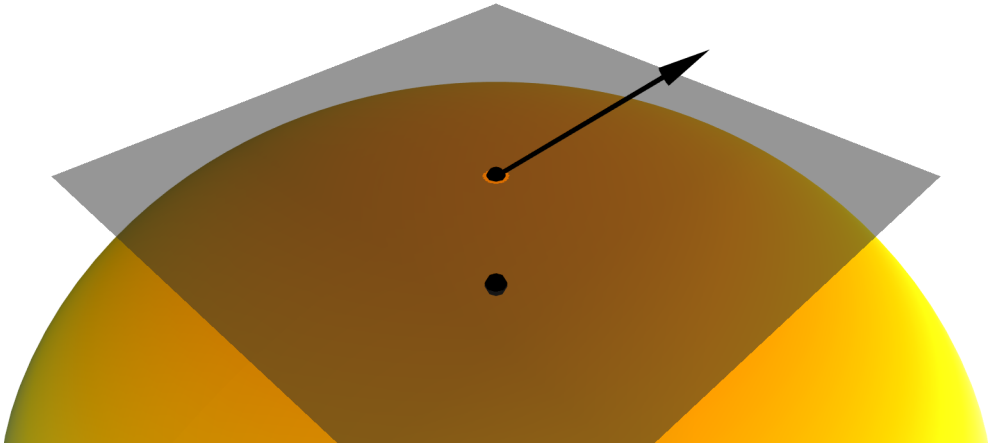
Step 4: Represent the tangent space $\mathcal{T}\mathcal{M}_r$ as $\dot{f} = \sum_{ij} \left(\dot{X}_i S_{ij} V_j + X_i \dot{S}_{ij} V_j + X_i S_{ij} \dot{V}_j \right)$.



Dynamical low-rank approximation

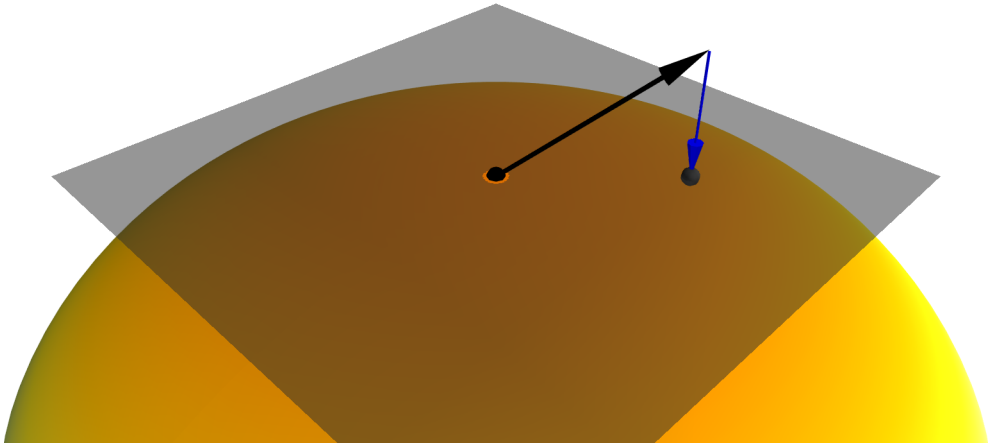
Step 4: Represent the tangent space $\mathcal{TM}_r$ as $\dot{f} = \sum_{ij} \left(\dot{X}_i S_{ij} V_j + X_i \dot{S}_{ij} V_j + X_i S_{ij} \dot{V}_j \right)$

- A point in the tangent space is specified by $\dot{X}_i, \dot{S}_{ij}, \dot{V}_j$.



Dynamical low-rank approximation

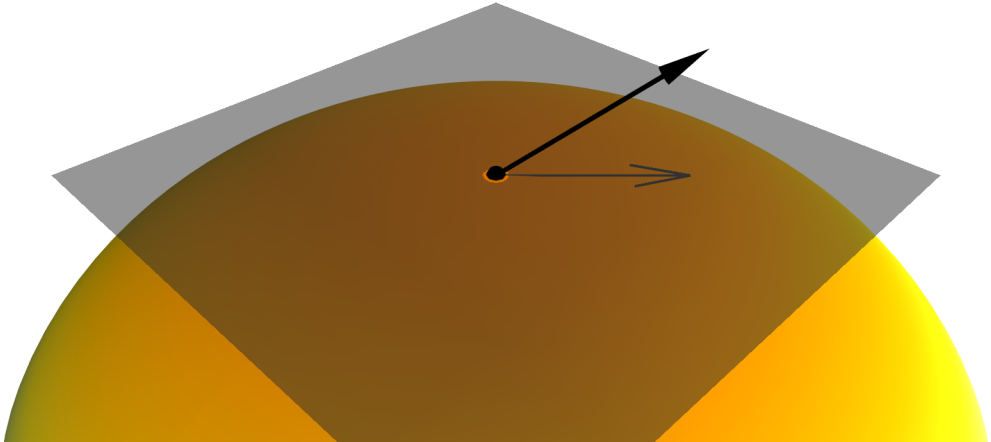
Step 5: Look for $\partial_t f = P(f)\mathrm{RHS}(f)$ with $P(f)$ the **orthogonal projection on the tangent space**.



Dynamical low-rank approximation

Step 5: Look for $\partial_t f = P(f)\mathrm{RHS}(f)$ with $P(f)$ the **orthogonal projection on the tangent space**.

- Determine $\dot{X}_i, \dot{S}_{ij}, \dot{V}_j$ from $\sum_{ij} \left(\dot{X}_i S_{ij} V_j + X_i \dot{S}_{ij} V_j + X_i S_{ij} \dot{V}_j \right) = P(f)\mathrm{RHS}$.



Dynamical low-rank approximation

Dynamical low-rank approximation

$$f(t,x,v) = \sum_{ij} X_i(t,x) S_{ij}(t) V_j(t,v).$$

Lower-dimensional basis is allowed to **vary in time**

Low-rank functions (with fixed r) form a **manifold** with functions in the tangent space represented as

$$\dot{f} = \sum_{ij} \left(\dot{X}_i S_{ij} V_j + X_i \dot{S}_{ij} V_j + X_i S_{ij} \dot{V}_j \right).$$

This representation is **not unique**. For example,

$$\dot{X}_i = X_i,\ \dot{S}_{ij} = 0 \qquad \text{and} \qquad \dot{X}_i = 0,\ \dot{S}_{ij} = S_{ij}$$

gives the same vector in the tangent space.

Gauge conditions

We impose the **Gauge conditions** $\langle X_i, \dot{X}_j \rangle_x = 0$ and $\langle V_i, \dot{V}_j \rangle_v = 0$.

Equation for S

$$\begin{aligned}
\langle X_k V_l, \dot{f} \rangle_{xv} &= \sum_{ij} \langle X_k V_l, \dot{X}_i S_{ij} V_j + X_i \dot{S}_{ij} V_j + X_i S_{ij} \dot{V}_j \rangle_{xv} \\
&= \sum_{ij} \langle X_k, \dot{X}_i \rangle_x S_{ij} \langle V_l, V_j \rangle_v + \sum_{ij} \langle X_k, X_i \rangle_x \dot{S}_{ij} \langle V_l, V_j \rangle_v + \sum_{ij} \langle X_k, X_i \rangle_x S_{ij} \langle V_l, \dot{V}_j \rangle_v \\
&= \dot{S}_{kl}
\end{aligned}$$

Equation for X

$$\begin{aligned}
\langle V_l, \dot{f} \rangle_v &= \sum_{ij} \langle V_l, \dot{X}_i S_{ij} V_j + X_i \dot{S}_{ij} V_j + X_i S_{ij} \dot{V}_j \rangle_v \\
&= \sum_{ij} \dot{X}_i S_{ij} \langle V_l, V_j \rangle_v + \sum_{ij} X_i \dot{S}_{ij} \langle V_l, V_j \rangle_v + \sum_{ij} X_i S_{ij} \langle V_l, \dot{V}_j \rangle_v \\
&= \sum_{i} \dot{X}_i S_{il} + \sum_{i} X_i \dot{S}_{il}
\end{aligned}$$

Dynamical low-rank approximation

Equations of motion

$$\partial_t S_{ij} = \langle X_i V_j, \mathrm{RHS} \rangle_{xv}, \qquad \text{ODE}$$

$$\sum_i S_{ij}(\partial_t X_i) = \langle V_j, \mathrm{RHS} \rangle_v - \sum_i X_i(\partial_t S_{ij}), \qquad \text{x dependent PDE}$$

$$\sum_j S_{ij}(\partial_t V_j) = \langle X_i, \mathrm{RHS} \rangle_x - \sum_j (\partial_t S_{ij}) V_j. \qquad \text{v dependent PDE}$$

We have obtained

- A set of partial PDEs for the low-rank factors X, S, V.
- The low-rank factors only depend on x, v but not on both (i.e. $X_i(t,x)$, $V_j(t,v)$ vs $f(t,x,v)$).
- Up to now independent of the specific PDE we want to solve.

Equations of motion for Vlasov–Poisson

We can now substitute

$$\text{RHS} = -v \cdot \nabla_x f + E(f) \cdot \nabla_v f \qquad \text{with} \qquad f = \sum_{kl} X_k S_{kl} V_l$$

For example,

$$\begin{aligned}\langle V_j, \text{RHS} \rangle_v &= \langle V_j, v \mapsto -v \cdot \nabla_x f + E(f) \cdot \nabla_v f \rangle_v \\ &= -\sum_l \langle V_j v V_l \rangle_v \cdot \sum_k S_{kl} \nabla_x X_k + \sum_l E \cdot \langle V_j \nabla_v V_l \rangle_v \sum_k S_{kl} \nabla_x X_k\end{aligned}$$

We can now compute this efficiently.

- Your choice of time and space discretization

Computational complexity

Memory usage: $\mathcal{O}(rn^d)$ instead of $\mathcal{O}(n^{d_x+d_v})$.

- Limited by storage of X_i and V_j.

Computational complexity: $\mathcal{O}(r^2n^d)$ instead of $\mathcal{O}(n^{d_x+d_v})$.

- Limited by computation of the coefficients and solution of evolution equations.

Coefficients:	$c_{jl}^1 = \int_{\Omega_v} vV_jV_l \,\mathrm{d}v$	Storage: $\mathcal{O}(r^2)$	Effort: $\mathcal{O}(r^2n^{d_v})$
Integration:	$\partial_t X_j = \ldots$	Storage: $\mathcal{O}(rn^{d_x})$	Effort: $\mathcal{O}(r^2n^{d_x})$

Dramatic improvement if r is small to moderate.

- E.g. $d_x = d_v = 3$, $n = 200$, $r = 10$ we have 1024 TB vs 160 MB to store two copies of f.

$d = \max(d_x, d_v)$.

Robustness

In principle, **we can now forget about low-rank** and **implement the new PDEs.**

But to obtain equations in X_i and V_j we have to invert S and S^{T} ($S\partial_t X = \dots$).

Approximation by truncation of the SVD $A \approx VSW^T$.

$$S = \begin{bmatrix} \mu_1 & 0 & 0 & 0 & 0 \\ 0 & \mu_2 & 0 & 0 & 0 \\ 0 & 0 & \mu_3 & 0 & 0 \\ 0 & 0 & 0 & \mu_4 & 0 \\ 0 & 0 & 0 & 0 & \mu_5 \end{bmatrix} \approx \begin{bmatrix} \mu_1 & 0 & 0 \\ 0 & \mu_2 & 0 \\ 0 & 0 & \mu_3 \end{bmatrix}.$$

Error vs condition number

- If μ_4 is large than the error is large.
- If μ_3 is small than inverting S is ill-conditioned.

Literature

Literature

[O. Koch, C. Lubich. SIAM J. Matrix Anal. Appl., 29(2), 2007]

- Dynamical low-rank algorithm for matrix equations, modern mathematical formulation.

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

- Projector splitting based dynamical low-rank algorithm for Vlasov–Poisson.
- Probably the best starting point to get more details.

[L.E., K. Kormann, J. Kusch, R.G. McClarren, J.-M. Qiu. arXiv:2412.05912]

- Review article with comparison to other methods and survey of the literature.