Numerical methods for plasma sheaths

CEMRACS 2022

Team: Valentin Ayot; Averil Prost; Christian Tayou Fotso.

Supervisors: Mehdi Badsi; Yann Barsamian; Anaïs Crestetto; Nicolas Crouseilles; Michel Mehrenberger.

June 1, 2023

CEMRACS 22

Table of Contents

Introduction

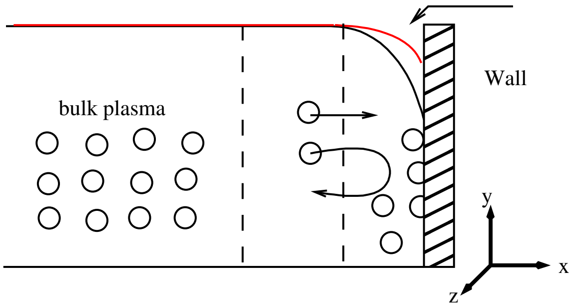
Building blocks
- Advection
- Poisson

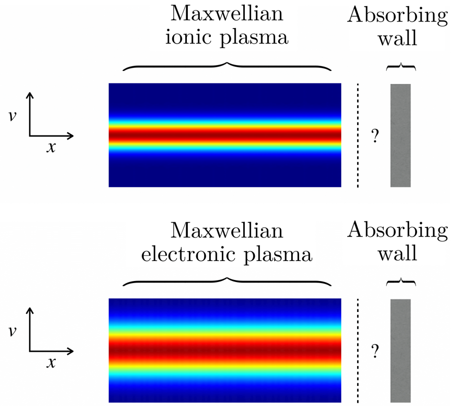
Algorithms for the full model
- Evolution problem
- Stationary problem
- Towards the equilibrium

Results
- Comparison between the evolutionary algorithms
- Dialog with equilibrium code

Motivation

Study the formation of steady *sheath*.





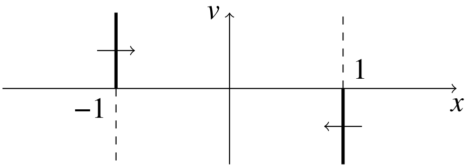
The model

$$
\begin{cases}
\partial_t f_i(t,x,v) + v\partial_x f_i(t,x,v) - \partial_x\phi(t,x)\partial_v f_i(t,x,v) = \nu f_e(t,x,v), \\
\partial_t f_e(t,x,v) + v\partial_x f_e(t,x,v) + \dfrac{1}{\mu}\partial_x\phi(t,x)\partial_v f_e(t,x,v) = 0, \\
-\lambda^2\partial_{xx}\phi(t,x) = \displaystyle\int_{\mathbb{R}} f_i(t,x,v)dv - \int_{\mathbb{R}} f_e(t,x,v)dv
\end{cases}
\tag{1}
$$

- $f_i := f_i(t,x,v)$, $(t,x,v) \in \mathbb{R}^+ \times [-1,1] \times \mathbb{R}$ Density of ions.
- $f_e := f_e(t,x,v)$, $(t,x,v) \in \mathbb{R}^+ \times [-1,1] \times \mathbb{R}$ Density of electrons.
- $\phi := \phi(t,x)$, $(t,x) \in \mathbb{R}^+ \times [-1,1]$ Electrostatic potential.
- mass ratio $\mu > 0$, ionization frequency $\nu \geqslant 0$, and Debye length $\lambda > 0$.

Boundary conditions

- $f_{i,e}(0, x, v) = f_{i,e}^0(x, v)$,
- $f_{i,e}(t, x = -1, v > 0) = 0$ and $f_{i,e}(t, x = 1, v < 0) = 0$,
- $\phi(t, 0) = 0$, $\partial_x \phi(t, 0) = 0$,
- $-\lambda^2 \partial_{xx}^2 \phi(0, x) = \int_{\mathbb{R}} f_i^0(x, v) dv - \int_{\mathbb{R}} f_e^0(x, v) dv.$



Boundary conditions

We consider

$$f_i^0(x,v) := \frac{e^{-v^2/2}}{\sqrt{2\pi}}, \quad f_e^0(x,v) := \frac{\sqrt{\mu}e^{-\mu v^2/2}}{\sqrt{2\pi}}, \quad \phi(0,x) = E(0,x) := 0.$$

Let the electric field and currents be defined $\forall (t,x) \in \mathbb{R}^+ \times [-1,1]$ as

$$E(t,x) := -\partial_x \phi(t,x), \quad \text{and} \quad J_{i,e}(t,x) := \int_{\mathbb{R}} v f_{i,e}(t,x,v) dv.$$

Then, denoting $J := J_i - J_e$,

$$\lambda^2 \partial_t E(t,\pm 1) := J(t,\pm 1) \pm \frac{\nu}{2} \int_{-1}^{1} \int_{\mathbb{R}} f_e(t,y,v)\, dy dv.$$

Objectives of the CEMRACS project

- Investigate numerically the stationary solution with high-order approximation method.
- Integrate nonperiodic boundary conditions.
- Tools to approximate Vlasov equations:
  - Finite difference code (FD),
  - Semi-Lagrangian code (SL),
  - Fixed-point code (FP).

Table of Contents

Introduction

Building blocks
- Advection
- Poisson

Algorithms for the full model
- Evolution problem
- Stationary problem
- Towards the equilibrium

Results
- Comparison between the evolutionary algorithms
- Dialog with equilibrium code

1D Transport

$$\begin{cases} \partial_t u(t,x) + c\partial_x u(t,x) &= 0 \quad \text{for } t \geq 0,\ x \in [a,b],\ c > 0, \\ u(0,x) &= u_0(x) \quad \text{for } x \in [a,b], \\ u(t,a) &= u_L(t) \quad \text{for } t \geq 0. \end{cases} \tag{2}$$

The solution is constant along the characteristic lines:

$$u(t,x) = \begin{cases} u_0(x-ct) & \text{if } x-a \geq ct, \\ u_L\left(t - \dfrac{x-a}{c}\right) & \text{if } x-a \leq ct. \end{cases} \tag{3}$$

Let $x_i := a + i\Delta x$, $\Delta x > 0$, $x_N = b$ and $t_n := n\Delta t$, $\Delta t > 0$.

$$u_i^{n+1} \simeq u(t_{n+1}, x_i) = u(t_n, x_i - c\Delta t). \tag{4}$$

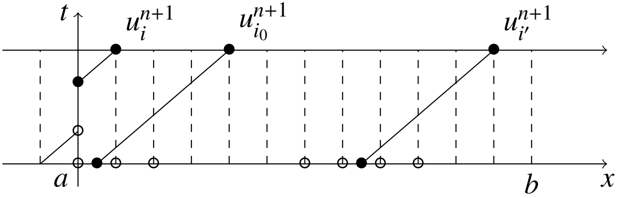
1D Transport

- Choice of Lagrange interpolation on a stencil of width $2d + 2$.
- Ghost points: Inflow x_{-i}, Outflow x_{N+i}, $i \in \mathbb{N}$.

Inflow: we set $i_0 := \lceil c\Delta t/\Delta x \rceil$, then for all $i < i_0$,

$$u_i^{n+1} \simeq u_L(t_{n+1} - i\Delta x/c). \tag{5}$$

For $i \in [\![i_0, d-1]\!]$, interpolation with $u_{-j}^n := u_L(t_n + j\Delta x/c)$, $j \in \mathbb{N}$.



1D Transport

Outflow: if the interpolation stencil needs ghost points at the right of the domain ($i_0 \leqslant d$).
Idea from (Coulombel *et al.* 2020):

- Introduce the finite difference operator $(Du)_j := u_j - u_{j-1}$.
- Let $k_b \in \mathbb{N}$, and deduce u^n_{N+i} by enforcing $\left(D^{k_b} u^n\right)_{N+i} = 0$, $i \in [\![1, d]\!]$.

Then u^n_{N+i} is a linear combination of $u^n_{N+i-1}, \cdots, u^n_{N+i-k_b}$.

Equivalent to polynomial extrapolation of order $k_b - 1$.

Poisson equation

Let $x \in [-1, 1]$. We look for symmetric solutions: $\phi(t, x) = \phi(t, -x)$, with

$$\begin{cases} -\lambda^2 \partial_{xx}^2 \phi(t, x) &= n(t, x) := \int_{v \in \mathbb{R}} (f_i - f_e)(t, x, v) dv, \\ \phi(t, 0) = \partial_x \phi(t, 0) &= 0. \end{cases} \tag{6}$$

Let $E := -\partial_x \phi$ the electric field in (1). Then $E(t, x) = -E(t, -x)$, with

$$\begin{cases} \lambda^2 \partial_x E(t, x) &= n(t, x) \\ -E(t, 0) &= 0. \end{cases} \implies E(t, x) = \frac{1}{\lambda^2} \int_0^x n(t, y) dy. \tag{7}$$

- Resolution by quadrature.
- Numerically, useful to enforce the symmetry of n.

Table of Contents

Introduction

Building blocks
- Advection
- Poisson

Algorithms for the full model
- Evolution problem
- Stationary problem
- Towards the equilibrium

Results
- Comparison between the evolutionary algorithms
- Dialog with equilibrium code

Finite difference discretization (FD, C++)

Classical explicit upwind scheme for f_s, $s \in \{e, i\}$:

$$\frac{f_{s,k,l}^{n+1} - f_{s,k,l}^{n}}{\Delta t} + \begin{pmatrix} v_l \\ c_s E_k^n \end{pmatrix}_+ D_{k,l}^- f_s^n + \begin{pmatrix} v_l \\ c_s E_k^n \end{pmatrix}_- D_{k,l}^+ f_s^n = S^n_{s,k,l}$$

where $S_e := 0$, $S_i := \nu f_e$, $c_e := -\frac{1}{\mu}$, $c_i := 1$, and $D_{k,l}^{\pm} f := \pm \left(\frac{f_{k\pm1,l} - f_{k,l}}{\Delta x}, \frac{f_{k,l\pm1} - f_{k,l}}{\Delta v} \right)^t$.

Finite difference discretization (FD, C++)

Classical explicit upwind scheme for f_s, $s \in \{e, i\}$:

$$\frac{f_{s,k,l}^{n+1} - f_{s,k,l}^{n}}{\Delta t} + \begin{pmatrix} v_l \\ c_s E_k^n \end{pmatrix}_+ D_{k,l}^- f_s^n + \begin{pmatrix} v_l \\ c_s E_k^n \end{pmatrix}_- D_{k,l}^+ f_s^n = S^{\,n}_{s,k,l}$$

where $S_e := 0$, $S_i := \nu f_e$, $c_e := -\frac{1}{\mu}$, $c_i := 1$, and $D_{k,l}^{\pm} f := \pm\left(\frac{f_{k\pm1,l} - f_{k,l}}{\Delta x}, \frac{f_{k,l\pm1} - f_{k,l}}{\Delta v}\right)^t$.
Integration for E, assuming $E(t, 0) = 0$:

$$E_k^{n+1} = \frac{1}{\lambda^2} \text{Trapezoid}_{\Delta x}\left(n_i^n - n_e^n\right)$$

Finite difference discretization (FD, C++)

Classical explicit upwind scheme for f_s, $s \in \{e, i\}$:

$$\frac{f_{s,k,l}^{n+1} - f_{s,k,l}^{n}}{\Delta t} + \begin{pmatrix} v_l \\ c_s E_k^n \end{pmatrix}_+ D_{k,l}^- f_s^n + \begin{pmatrix} v_l \\ c_s E_k^n \end{pmatrix}_- D_{k,l}^+ f_s^n = S^{\,n}_{s,k,l}$$

where $S_e := 0$, $S_i := \nu f_e$, $c_e := -\frac{1}{\mu}$, $c_i := 1$, and $D_{k,l}^{\pm} f := \pm \left(\frac{f_{k\pm1,l} - f_{k,l}}{\Delta x}, \frac{f_{k,l\pm1} - f_{k,l}}{\Delta v} \right)^t$.
Integration for E, assuming $E(t, 0) = 0$:

$$E_k^{n+1} = \frac{1}{\lambda^2} \text{Trapezoid}_{\Delta x} \left(n_i^n - n_e^n \right)$$

- First order, diffusive, CFL condition.

Finite difference discretization (FD, C++)

Classical explicit upwind scheme for f_s, $s \in \{e, i\}$:

$$\frac{f^{n+1}_{s,k,l} - f^n_{s,k,l}}{\Delta t} + \begin{pmatrix} v_l \\ c_s E^n_k \end{pmatrix}_+ D^-_{k,l} f^n_s + \begin{pmatrix} v_l \\ c_s E^n_k \end{pmatrix}_- D^+_{k,l} f^n_s = S^{\,n}_{s,k,l}$$

where $S_e := 0$, $S_i := \nu f_e$, $c_e := -\frac{1}{\mu}$, $c_i := 1$, and $D^{\pm}_{k,l} f := \pm \left(\frac{f_{k\pm1,l} - f_{k,l}}{\Delta x}, \frac{f_{k,l\pm1} - f_{k,l}}{\Delta v} \right)^t$.
Integration for E, assuming $E(t, 0) = 0$:

$$E^{n+1}_k = \frac{1}{\lambda^2} \text{Trapezoid}_{\Delta x} \left(n^n_i - n^n_e \right)$$

- First order, diffusive, CFL condition.
- Common speed mesh for electrons and ions.

Semi-Lagrangian scheme (SL, C)

Strang splitting:

$$\frac{\Delta t}{2} \qquad \begin{cases} \partial_t f_s + v \partial_x f_s = 0 & \text{Linear advection at constant speed} \\ \lambda^2 \partial_x E = n_i - n_e & \text{Resolution by (numerical) integration} \end{cases}$$

$$\frac{\Delta t}{2} \qquad \begin{cases} \partial_t f_i = \nu f_e & \text{Pointwise ODE} \end{cases}$$

Semi-Lagrangian scheme (SL, C)

Strang splitting:

$\frac{\Delta t}{2}$ $\begin{cases} \partial_t f_s + v\partial_x f_s = 0 \\ \lambda^2 \partial_x E = n_i - n_e \end{cases}$ Linear advection at constant speed / Resolution by (numerical) integration

$\frac{\Delta t}{2}$ $\begin{cases} \partial_t f_i = \nu f_e \end{cases}$ Pointwise ODE

Δt $\partial_t f_s + c_s E \partial_v f_s = 0$ Again, advection at constant speed

Semi-Lagrangian scheme (SL, C)

Strang splitting:

$$\frac{\Delta t}{2} \quad \begin{cases} \partial_t f_s + v\partial_x f_s = 0 & \text{Linear advection at constant speed} \\ \lambda^2 \partial_x E = n_i - n_e & \text{Resolution by (numerical) integration} \end{cases}$$

$$\frac{\Delta t}{2} \quad \begin{cases} \partial_t f_i = \nu f_e & \text{Pointwise ODE} \end{cases}$$

$$\Delta t \quad \partial_t f_s + c_s E \partial_v f_s = 0 \quad \text{Again, advection at constant speed}$$

$$\frac{\Delta t}{2} \quad \begin{cases} \partial_t f_i = \nu f_e & \text{Pointwise ODE} \end{cases}$$

$$\frac{\Delta t}{2} \quad \begin{cases} \lambda^2 \partial_x E = n_i - n_e & \text{Resolution by (numerical) integration} \\ \partial_t f_s + v\partial_x f_s = 0 & \text{Linear advection at constant speed.} \end{cases}$$

Semi-Lagrangian scheme (SL, C)

Strang splitting:

$$\frac{\Delta t}{2} \qquad \begin{cases} \partial_t f_s + v\partial_x f_s = 0 & \text{Linear advection at constant speed} \\ \lambda^2 \partial_x E = n_i - n_e & \text{Resolution by (numerical) integration} \end{cases}$$

$$\frac{\Delta t}{2} \qquad \begin{cases} \partial_t f_i = \nu f_e & \text{Pointwise ODE} \end{cases}$$

$$\Delta t \qquad \partial_t f_s + c_s E \partial_v f_s = 0 \quad \text{Again, advection at constant speed}$$

$$\frac{\Delta t}{2} \qquad \begin{cases} \partial_t f_i = \nu f_e & \text{Pointwise ODE} \end{cases}$$

$$\frac{\Delta t}{2} \qquad \begin{cases} \lambda^2 \partial_x E = n_i - n_e & \text{Resolution by (numerical) integration} \\ \partial_t f_s + v\partial_x f_s = 0 & \text{Linear advection at constant speed.} \end{cases}$$

- Use of the 1D solver (with appropriate boundary conditions).

Semi-Lagrangian scheme (SL, C)

Strang splitting:

$$\frac{\Delta t}{2} \quad \begin{cases} \partial_t f_s + v\partial_x f_s = 0 & \text{Linear advection at constant speed} \\ \lambda^2 \partial_x E = n_i - n_e & \text{Resolution by (numerical) integration} \end{cases}$$

$$\frac{\Delta t}{2} \quad \begin{cases} \partial_t f_i = \nu f_e & \text{Pointwise ODE} \end{cases}$$

$$\Delta t \quad \partial_t f_s + c_s E \partial_v f_s = 0 \quad \text{Again, advection at constant speed}$$

$$\frac{\Delta t}{2} \quad \begin{cases} \partial_t f_i = \nu f_e & \text{Pointwise ODE} \end{cases}$$

$$\frac{\Delta t}{2} \quad \begin{cases} \lambda^2 \partial_x E = n_i - n_e & \text{Resolution by (numerical) integration} \\ \partial_t f_s + v\partial_x f_s = 0 & \text{Linear advection at constant speed.} \end{cases}$$

- Use of the 1D solver (with appropriate boundary conditions).
- Different speed meshes for f_e and f_i.

Fixed-point algorithm (Badsi *et al.* 2021) (FP, Julia ♥)

Equilibrium state:

$$\begin{cases} v\partial_x f_s - c_s \partial_x \phi \partial_v f_s = S_s, \quad \frac{d}{d\tau}[f_s(x_s(\tau), v_s(\tau))] = S_s(x_s(\tau), v_s(\tau)) \\ -\lambda^2 \partial^2_{xx}\phi = n_i - n_e \end{cases}$$

Fixed-point algorithm (Badsi *et al.* 2021) (FP, Julia ♥)

Equilibrium state:

$$\begin{cases} v\partial_x f_s - c_s \partial_x \phi \partial_v f_s = S_s, & \frac{d}{d\tau}[f_s(x_s(\tau), v_s(\tau))] = S_s(x_s(\tau), v_s(\tau)) \\ -\lambda^2 \partial_{xx}^2 \phi = n_i - n_e \end{cases}$$

Suppose ϕ^k is known. Iteration of

$$f_e^k(x,v) \quad := \quad f_{e,b}(x_e(-\tau), v_e(-\tau))$$

Fixed-point algorithm (Badsi *et al.* 2021) (FP, Julia ♥)

Equilibrium state:

$$\begin{cases} v\partial_x f_s - c_s \partial_x \phi \partial_v f_s = S_s, & \frac{d}{d\tau}[f_s(x_s(\tau), v_s(\tau))] = S_s(x_s(\tau), v_s(\tau)) \\ -\lambda^2 \partial^2_{xx}\phi = n_i - n_e \end{cases}$$

Suppose ϕ^k is known. Iteration of

$$\begin{aligned} f_e^k(x,v) &:= f_{e,b}(x_e(-\tau), v_e(-\tau)) \\ f_i^k(x,v) &:= f_{i,b}(x_i(-\tau), v_i(-\tau)) + \int_{-\tau}^{0} v f_e^k(x_i(s), v_i(s)) ds \end{aligned}$$

Fixed-point algorithm (Badsi *et al.* 2021) (FP, Julia ♥)

Equilibrium state:

$$\begin{cases} v\partial_x f_s - c_s \partial_x \phi \partial_v f_s = S_s, & \frac{d}{d\tau}[f_s(x_s(\tau), v_s(\tau))] = S_s(x_s(\tau), v_s(\tau)) \\ -\lambda^2 \partial^2_{xx}\phi = n_i - n_e \end{cases}$$

Suppose ϕ^k is known. Iteration of

$$\begin{aligned} f_e^k(x,v) &:= f_{e,b}(x_e(-\tau), v_e(-\tau)) \\ f_i^k(x,v) &:= f_{i,b}(x_i(-\tau), v_i(-\tau)) + \int_{-\tau}^{0} \nu f_e^k(x_i(s), v_i(s))ds \\ -\lambda^2 \partial^2_{xx}\phi^{k+1} &:= \int_v [f_i^k(\cdot, v) - f_e^k(\cdot, v)]dv \end{aligned}$$

until convergence.

What does equilibrium look like

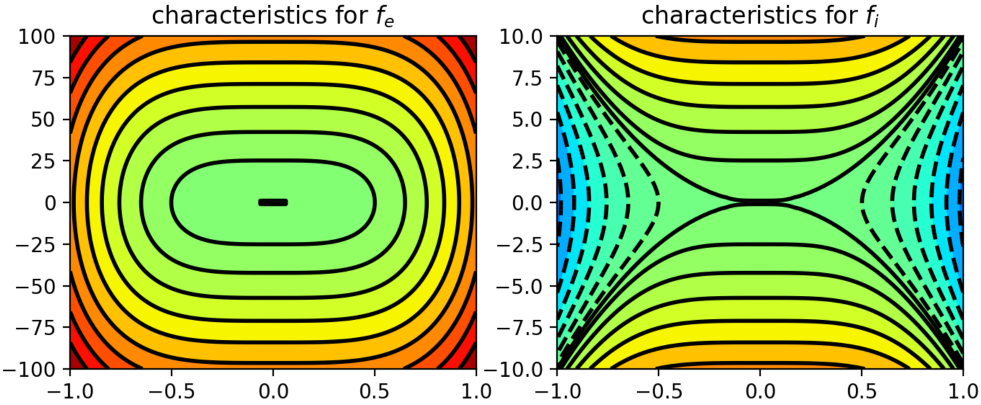
- The electron density $f_e(x, v)$ is constant along its characteristics.

What does equilibrium look like

- The electron density $f_e(x,v)$ is constant along its characteristics.
- The ion density $f_i(x,v)$ is the integral along its characteristics of vf_e.

What does equilibrium look like

- The electron density $f_e(x, v)$ is constant along its characteristics.
- The ion density $f_i(x, v)$ is the integral along its characteristics of $v f_e$.



What does equilibrium look like

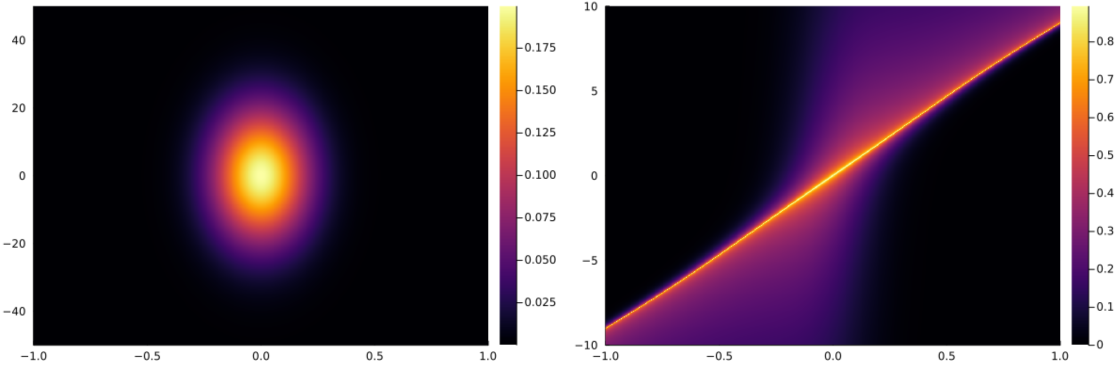


Figure: Left: electron density f_e. Right: ion density f_i (logscale).

$$\lambda = 0.1, \quad \mu := \frac{m_e}{m_i} = \frac{1}{100}, \quad \nu = 42, \quad N_{x,v_e,v_i} = 1024.$$

Table of Contents

Introduction

Building blocks
- Advection
- Poisson

Algorithms for the full model
- Evolution problem
- Stationary problem
- Towards the equilibrium

Results
- Comparison between the evolutionary algorithms
- Dialog with equilibrium code

Validation test case (DF): one-species (Malkov 2020)

Discretization parameters:

$$x \in [-1.5, 1.5],\ v_e \in [-2, 2],\ N_x = 2048,\ N_v = 2049.$$

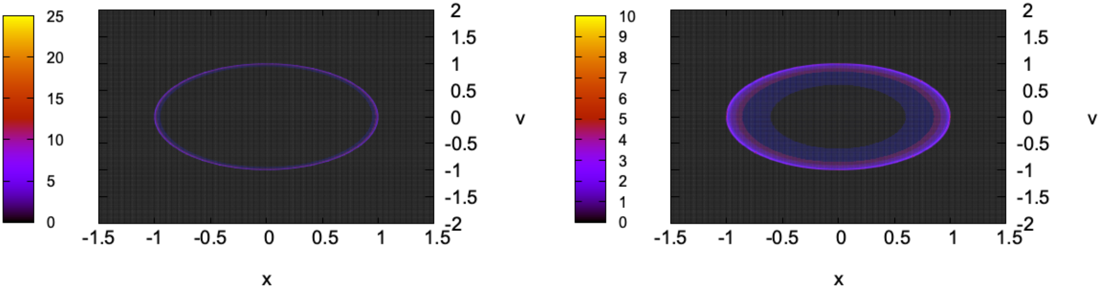


Figure: Left: solution at $t = 0.01$, Right: solution at $t = 0.05$.

Validation test case (DF): one-species (Malkov 2020)

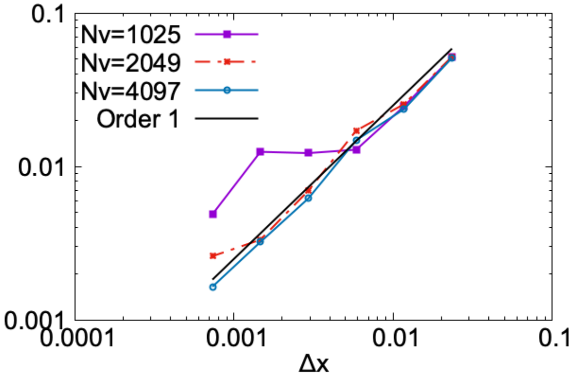


Figure: Error L^1 on the field E at time $T = 0.1$.

(SL)/(DF): Two-species Vlasov-Poisson

Simulation parameters:

$$\begin{cases} x \in [-1, 1],\ v_e \in [-20, 20],\ v_i \in [-10, 10],\ N_x = 256,\ N_{v_e} = 255, \\ N_{v_i} = 255,\ d = 8,\ k_b = 1,\ \mu = 1/100, \nu = 10,\ N_t = 3000,\ T = 3. \end{cases}$$

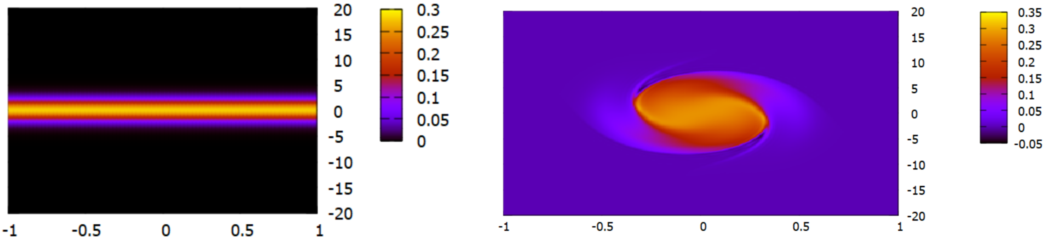
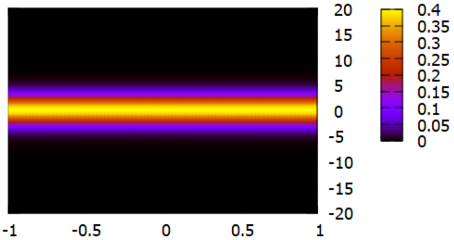


Figure: Electron distribution in phase space. Left: initial condition, right: time $T = 3$.

(SL)/(DF): Two-species Vlasov-Poisson

Simulation parameters:

$$\begin{cases} x \in [-1,1],\ v_e \in [-20,20],\ v_i \in [-10,10],\ N_x = 256,\ N_{v_e} = 255, \\ N_{v_i} = 255,\ d = 8,\ k_b = 1,\ \mu = 1/100, \nu = 10,\ N_t = 3000,\ T = 3. \end{cases}$$



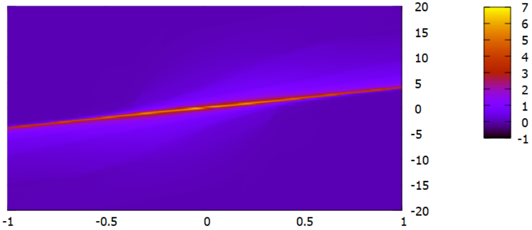


Figure: Ion distribution in phase space. Left: initial condition, right: time $T = 3$.

(SL)/(DF): Two-species Vlasov-Poisson with mask

Idea: multiply the initial conditions by a mask $\mathcal{M}$ defined as

$$\mathcal{M}(x) := \frac{1}{2}\left(\tanh\left(\frac{x-x_l}{d_r}\right) - \tanh\left(\frac{x-x_r}{d_r}\right)\right), \quad \text{with } x_l = -0.1,\ x_r = 0.1,\ d_r = 0.1.$$

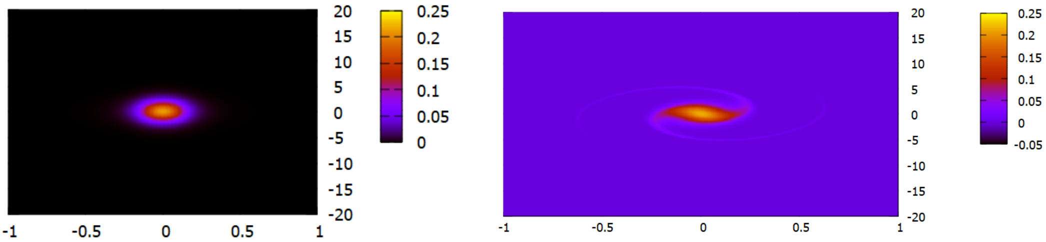
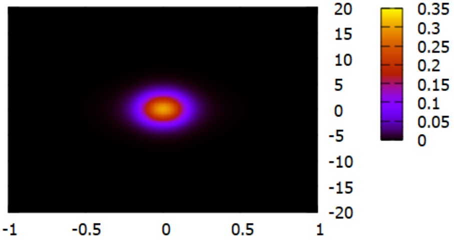


Figure: Electron distribution in phase space. Left: initial condition, right: time $T = 3$.

(SL)/(DF): Two-species Vlasov-Poisson with mask

Idea: multiply the initial conditions by a mask $\mathcal{M}$ defined as

$$\mathcal{M}(x) := \frac{1}{2}\left(\tanh\left(\frac{x-x_l}{d_r}\right) - \tanh\left(\frac{x-x_r}{d_r}\right)\right), \quad \text{with } x_l = -0.1,\ x_r = 0.1,\ d_r = 0.1.$$



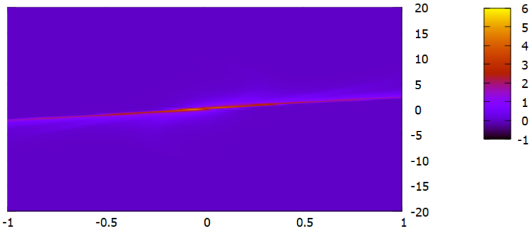


Figure: Ion distribution in phase space. Left: initial condition, right: time $T = 3$.

From (SL) to (FP)

Simulation parameters:

$$\begin{cases} x \in [-1,1],\ v_e \in [-10,10],\ v_i \in [-8,8], \\ N_x = 8192,\ N_{v_e} = 511,\ N_{v_i} = 4095, \\ \mu = 1/2,\ \nu = 20,\ \lambda = 0.5. \end{cases}$$

Initial conditions:

$$\begin{cases} f_e(t=0,x,v) & := \mathcal{M}(x) \times f_e^0(x,v), \\ f_i(t=0,x,v) & := \mathcal{M}(x) \times f_i^0(x,v). \end{cases}$$

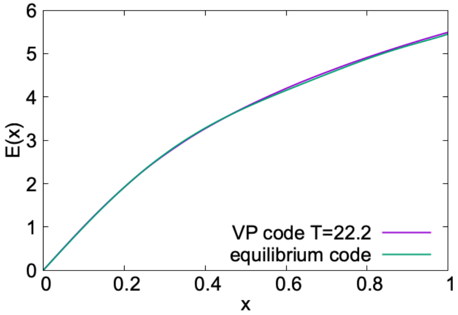


Figure: Electric field E at $T = 22.2$.

From (SL) to (FP)

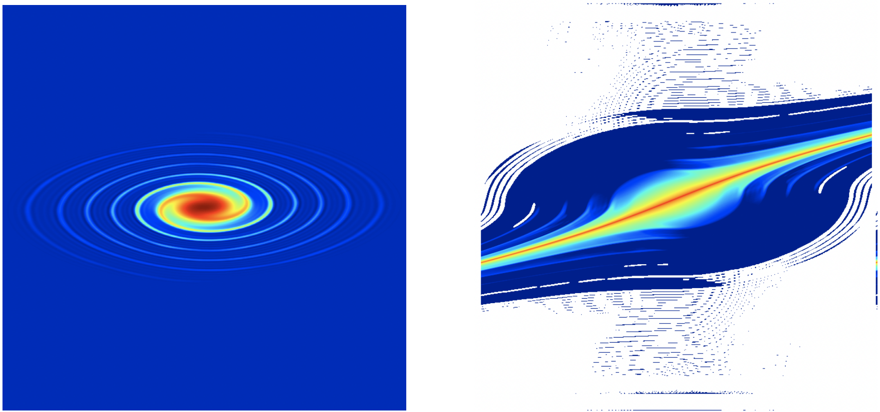


Figure: Electron and ion distributions in phase space. Left: f_e, right: f_i.

Equilibrium code

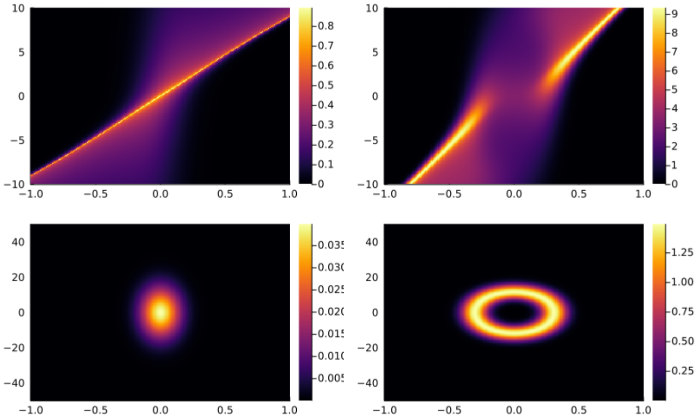


Figure: Equilibrium densities (up: f_i, down: f_e) for nonvanishing and vanishing $f_e(0,0)$.

From (FP) to (SL)

- Idea: initialize the semi-Lagrangian with the computed steady state.

From (FP) to (SL)

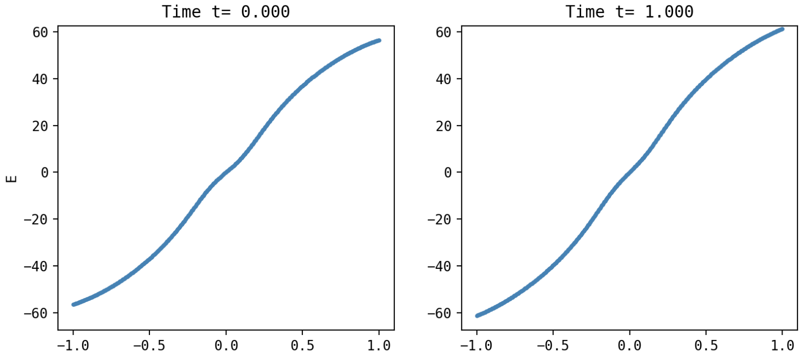
- Idea: initialize the semi-Lagrangian with the computed steady state.
- Expected: stationary state.

From (FP) to (SL)

- Idea: initialize the semi-Lagrangian with the computed steady state.
- Expected: stationary state.
- Result: some smoothing & oscillations, needs further investigations.

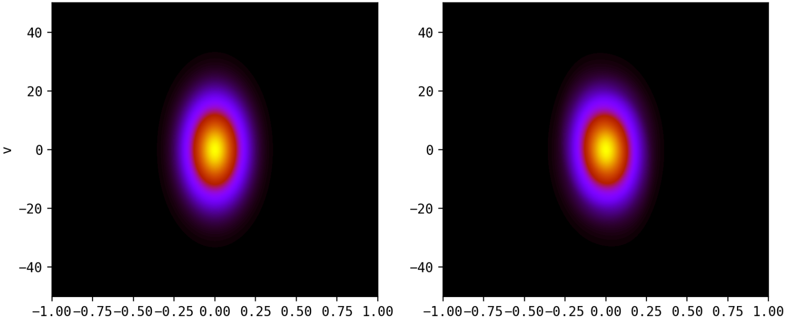
From (FP) to (SL)

- Idea: initialize the semi-Lagrangian with the computed steady state.
- Expected: stationary state.
- Result: some smoothing & oscillations, needs further investigations.



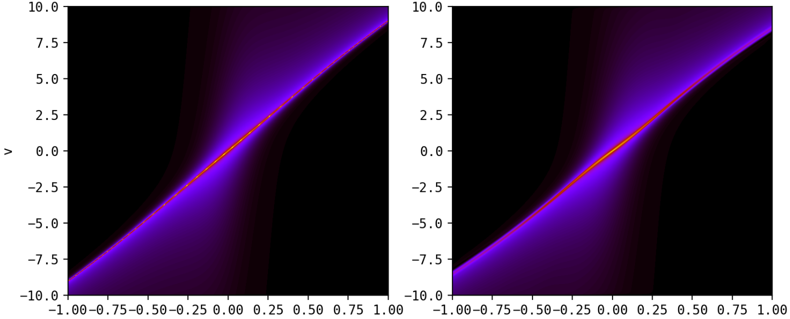
From (FP) to (SL)

- Idea: initialize the semi-Lagrangian with the computed steady state.
- Expected: stationary state.
- Result: some smoothing & oscillations, needs further investigations.



From (FP) to (SL)

- Idea: initialize the semi-Lagrangian with the computed steady state.
- Expected: stationary state.
- Result: some smoothing & oscillations, needs further investigations.



Conclusion and perspectives

What we did:

- Boundary conditions by extrapolation in a semi-Lagrangian algorithm.

Conclusion and perspectives

What we did:

- Boundary conditions by extrapolation in a semi-Lagrangian algorithm.
- Full Lagrangian fixed-point algorithm for the steady state.

Conclusion and perspectives

What we did:

- Boundary conditions by extrapolation in a semi-Lagrangian algorithm.
- Full Lagrangian fixed-point algorithm for the steady state.

What we may do:

- Test the equilibrium state in the FD algorithm (beware diffusion),

Conclusion and perspectives

What we did:

- Boundary conditions by extrapolation in a semi-Lagrangian algorithm.
- Full Lagrangian fixed-point algorithm for the steady state.

What we may do:

- Test the equilibrium state in the FD algorithm (beware diffusion),
- Representations of ϕ (polynomial, spectral, finite elements),

Conclusion and perspectives

What we did:

- Boundary conditions by extrapolation in a semi-Lagrangian algorithm.
- Full Lagrangian fixed-point algorithm for the steady state.

What we may do:

- Test the equilibrium state in the FD algorithm (beware diffusion),
- Representations of ϕ (polynomial, spectral, finite elements),
- Regularity of n_i, convergence of the fixed-point.

Conclusion and perspectives

What we did:

- Boundary conditions by extrapolation in a semi-Lagrangian algorithm.
- Full Lagrangian fixed-point algorithm for the steady state.

What we may do:

- Test the equilibrium state in the FD algorithm (beware diffusion),
- Representations of ϕ (polynomial, spectral, finite elements),
- Regularity of n_i, convergence of the fixed-point.

What we want:

- On behalf of everyone, thank the organisers and advisors for these amazing weeks.

Conclusion and perspectives

What we did:

- Boundary conditions by extrapolation in a semi-Lagrangian algorithm.
- Full Lagrangian fixed-point algorithm for the steady state.

What we may do:

- Test the equilibrium state in the FD algorithm (beware diffusion),
- Representations of ϕ (polynomial, spectral, finite elements),
- Regularity of n_i, convergence of the fixed-point.

What we want:

- On behalf of everyone, thank the organisers and advisors for these amazing weeks.

That's all folks, enjoy the BOUMLLABAISSE!

Thank you for your attention

Mehdi Badsi, Christophe Berthon, and Anaïs Crestetto.
A stable fixed point method for the numerical simulation of a kinetic collisional sheath.
Journal of Computational Physics, 429:109990, March 2021.

Jean-François Coulombel and Frédéric Lagoutière.
The Neumann numerical boundary condition for transport equations.
Kinetic & Related Models, 13(1):1–32, 2020.