HPC Master Class

Numerical modeling of kinetic plasmas

Nicolas Aunai

nicolas.aunai@lpp.polytechnique.fr

https://nicolasaunai.github.io/



Laboratory of Plasma Physics

HPC: High Performance Computing for Astrophysics

1- Numerical Modeling of Kinetic Plasmas

2- Numerical Modeling of Astrophysical Fluids

3- Machine Learning and GPU acceleration

Numerical Modeling of Kinetic Plasmas

- Astro-plasma-**physics**
- **Numerical** modeling for **kinetic** plasmas
- **Modern HPC**: C++ ecosystem (cmake, git, MPI, etc.)
- Hands-on: develop your own code

Evaluation?

- 2/3:
  - in-class **participation & autonomy**
  - Project : **code quality** and **correctness**
- 1/3:
  - **Oral** interview

Goals?

- Give you an **overview** of some basics
- Help you **determine your taste for astro-HPC**
- **Have fun** writing code for physics

Expect difficulties…



Class Outline

Session 1: plasmas and Coding

Session 2: Particle-In-Cell 1

Session 3: Particle-In-Cell 2

Session 4: Particle-In-Cell 3

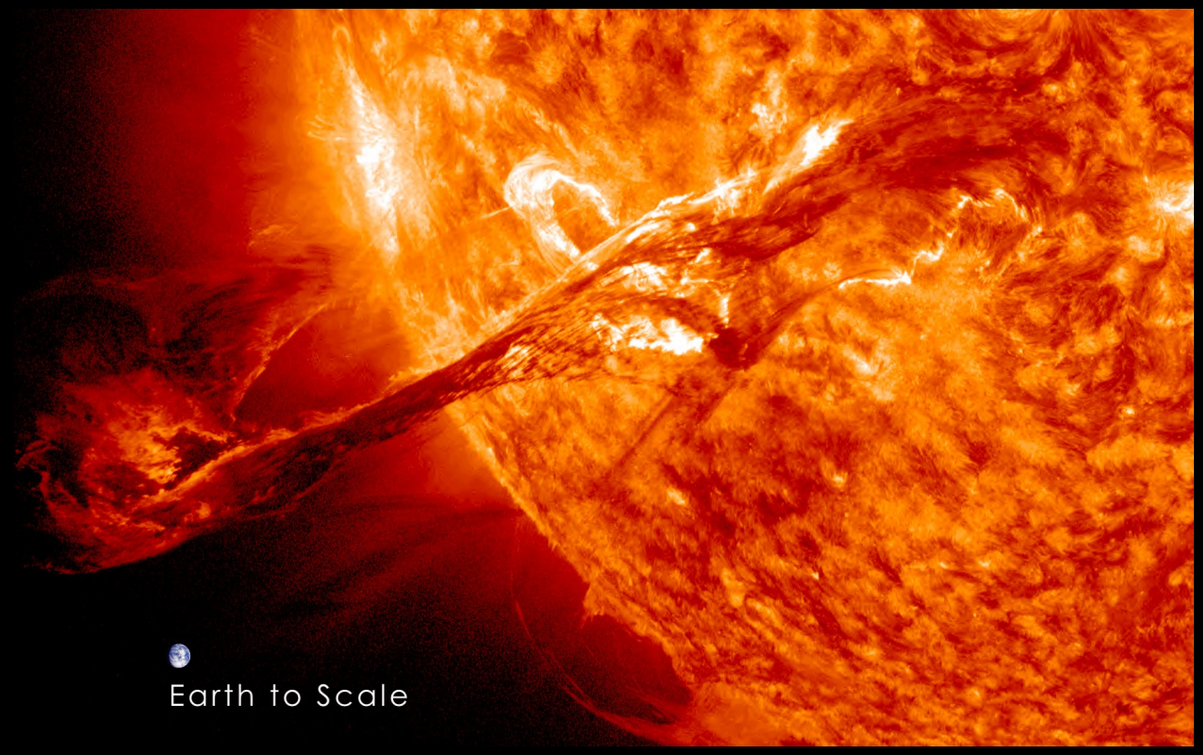
Session 5: Parallel Particle-In-Cell

Most of the blah-blah…

Most of the hands-on



SESSION 1: PLASMAS AND CODING









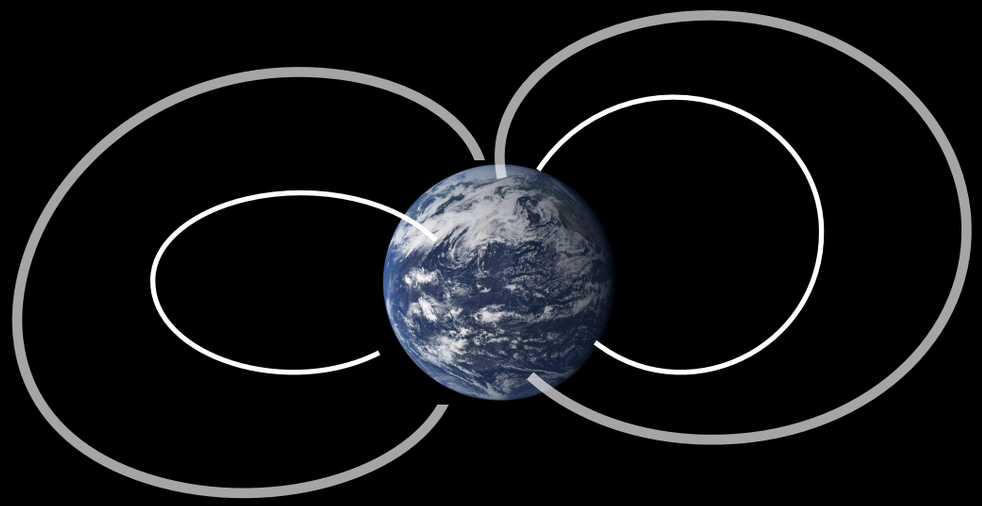


Plasma Physics Formalisms

- **This part of the class** is so you know **why we care** about numerical **kinetic modeling**
- It is an **overview** of the reasons
- You do not need to get all the details

- It is also a very interesting topic
- And what I do daily :)

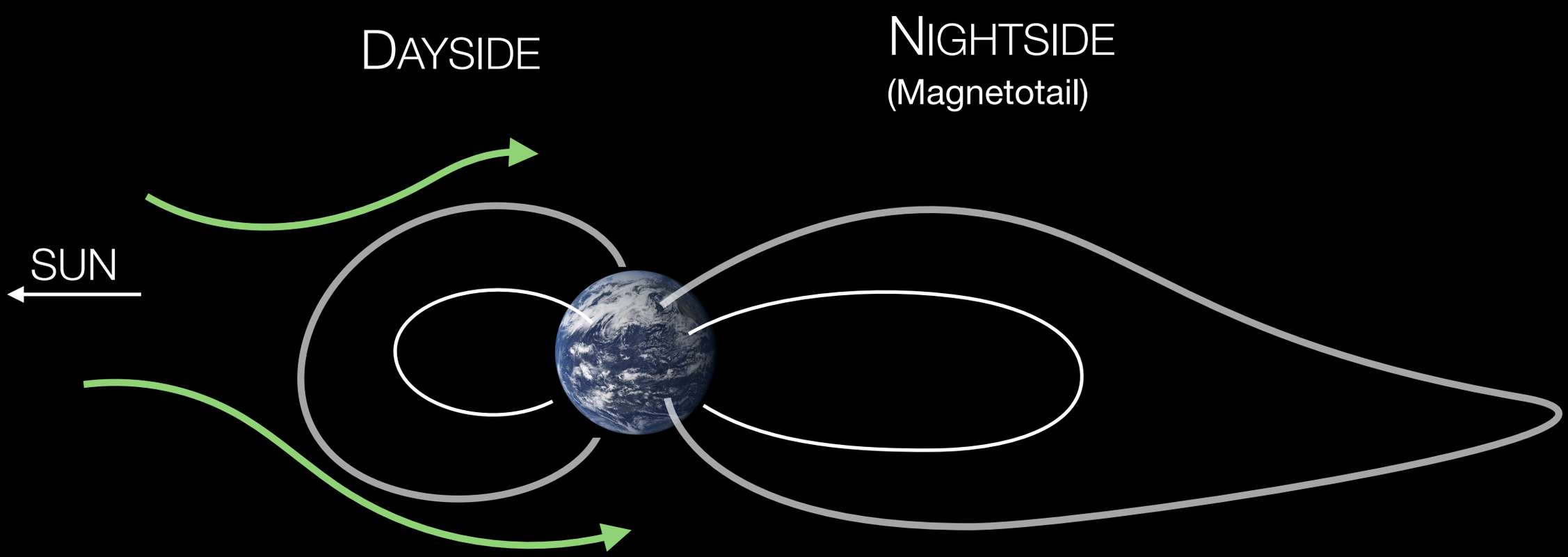
The Earth Magnetic Field



- The Earth has a **~dipolar magnetic field**
- It is generated by the **dynamo** effect
- It extends in space

- In vacuum the field extends to infinity
- But **space is not empty**… is it?

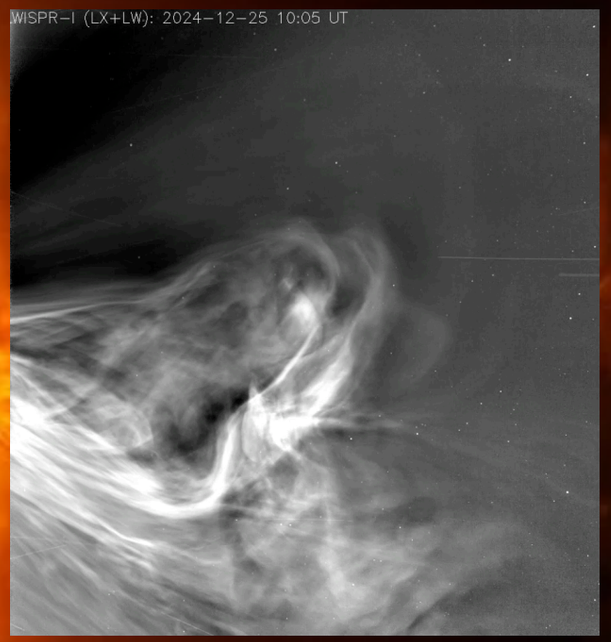
The Earth Magnetosphere



- The Sun blows radially a plasma wind: the **solar wind**
- The solar wind **compresses** the Earth magnetic field on the **dayside**
- And **stretches** it on the **nightside**

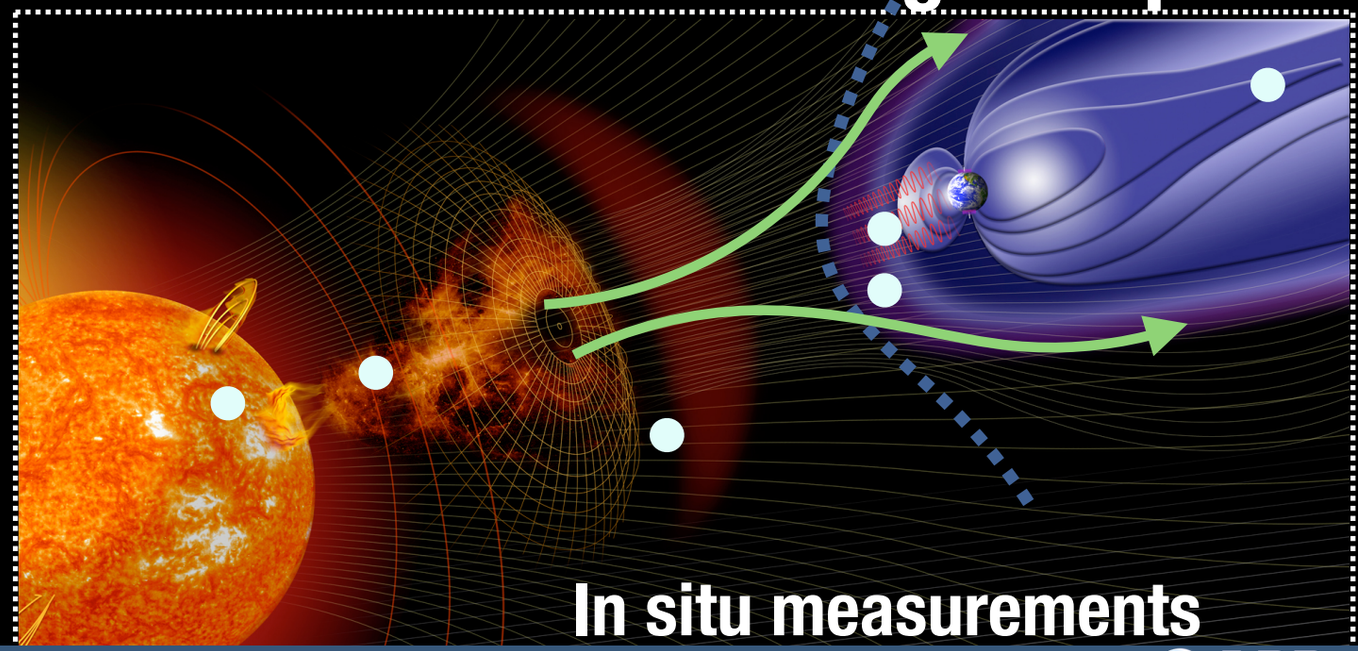
Plasma Universe

- The Sun-Earth (or Star-planet) interaction is a **complex field of physics**
- Many **interesting plasma processes** occur from the Sun to the Planet:
  - Convection zone (dynamo),
  - coronal dynamics (jets, eruptions, coronal mass ejections, etc.)
  - turbulent transport in the solar wind,
  - dayside magnetosphere interaction (reconnection, instabilities, etc.)
  - Inner magnetospheric dynamics

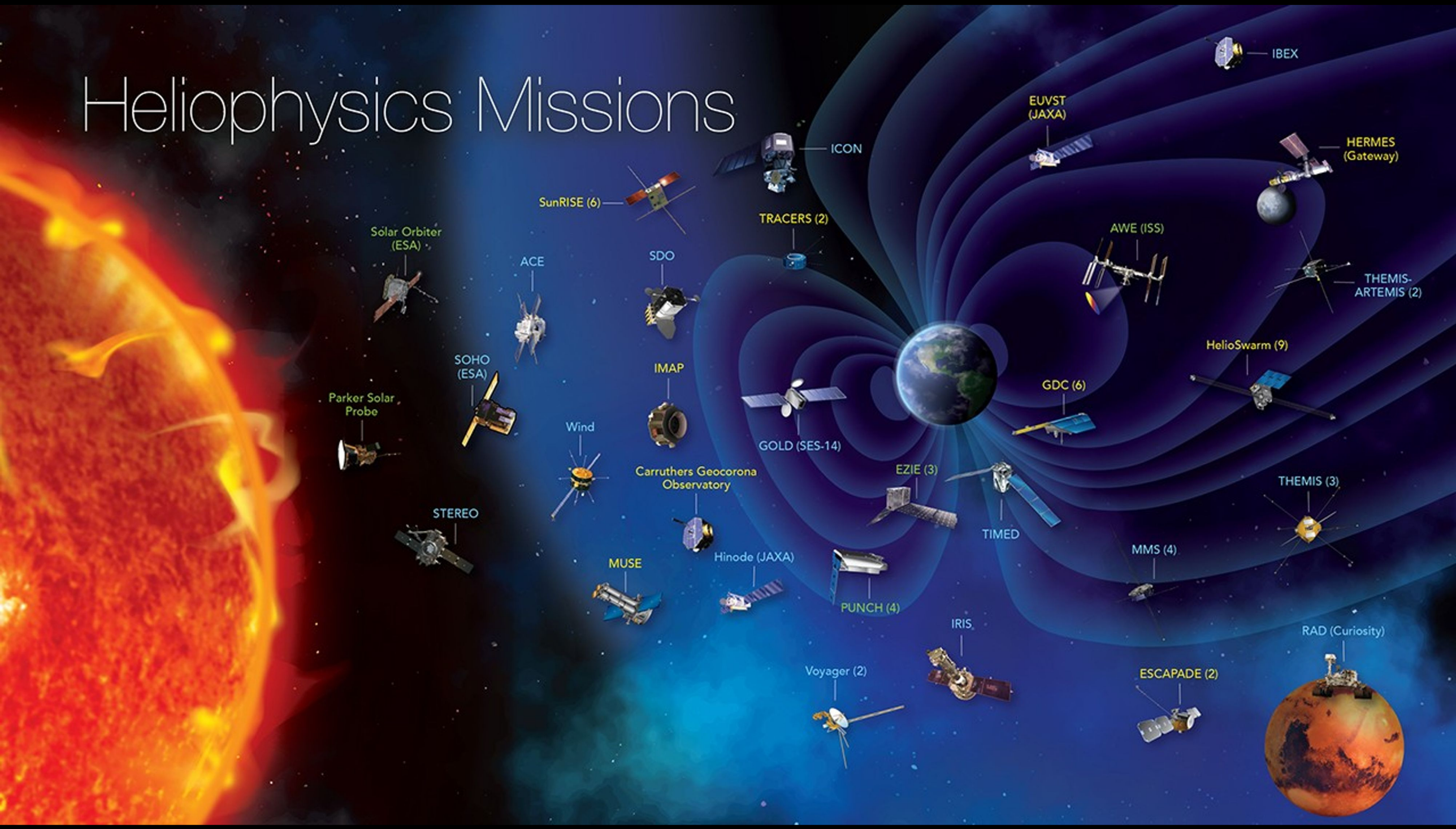


Universality of plasma processes

- The Sun-Earth system is a **unique laboratory** to explore all these processes
- It is relatively easily accessible to **in situ space measurements**



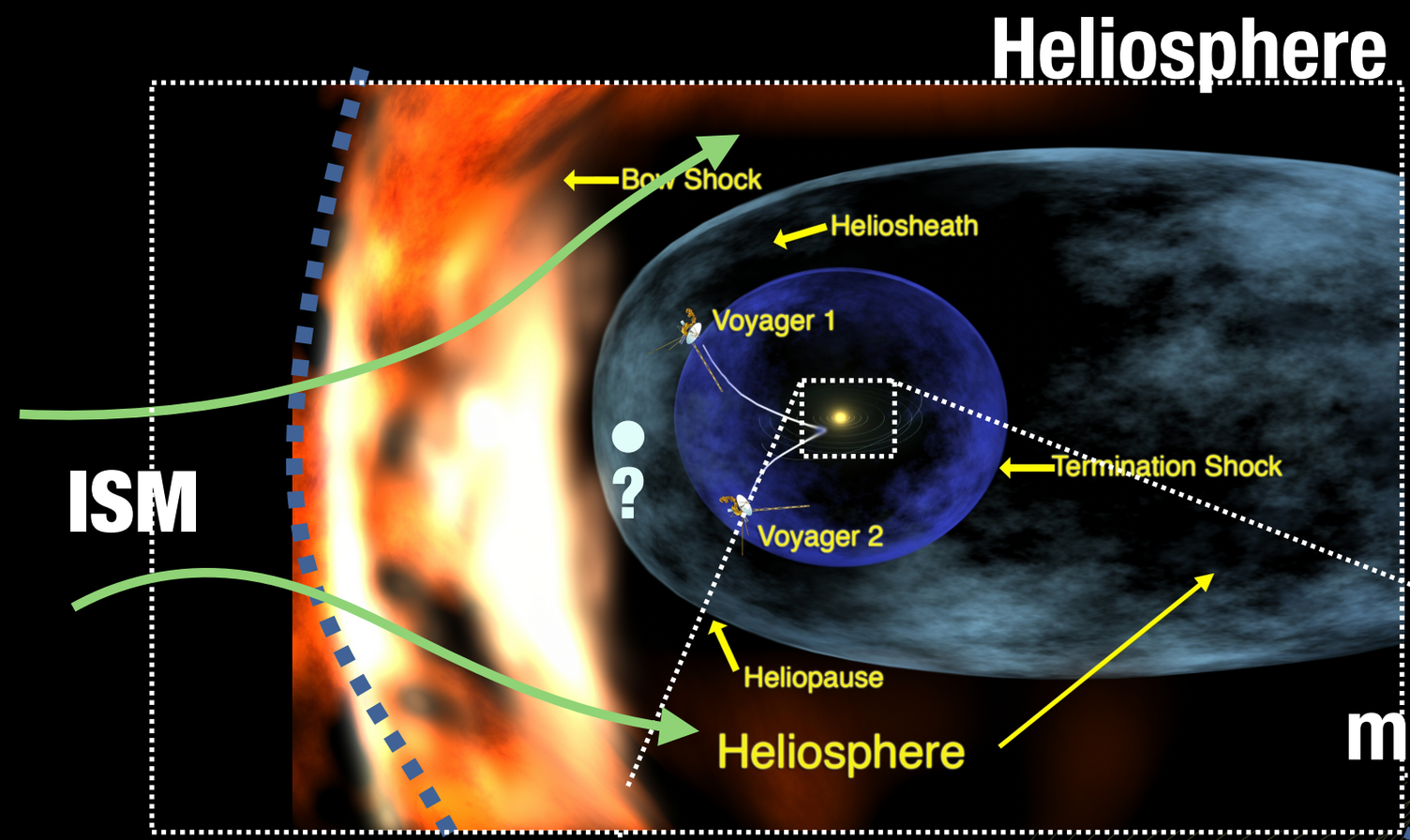
UNIVERSALITY OF PLASMA PROCESSES



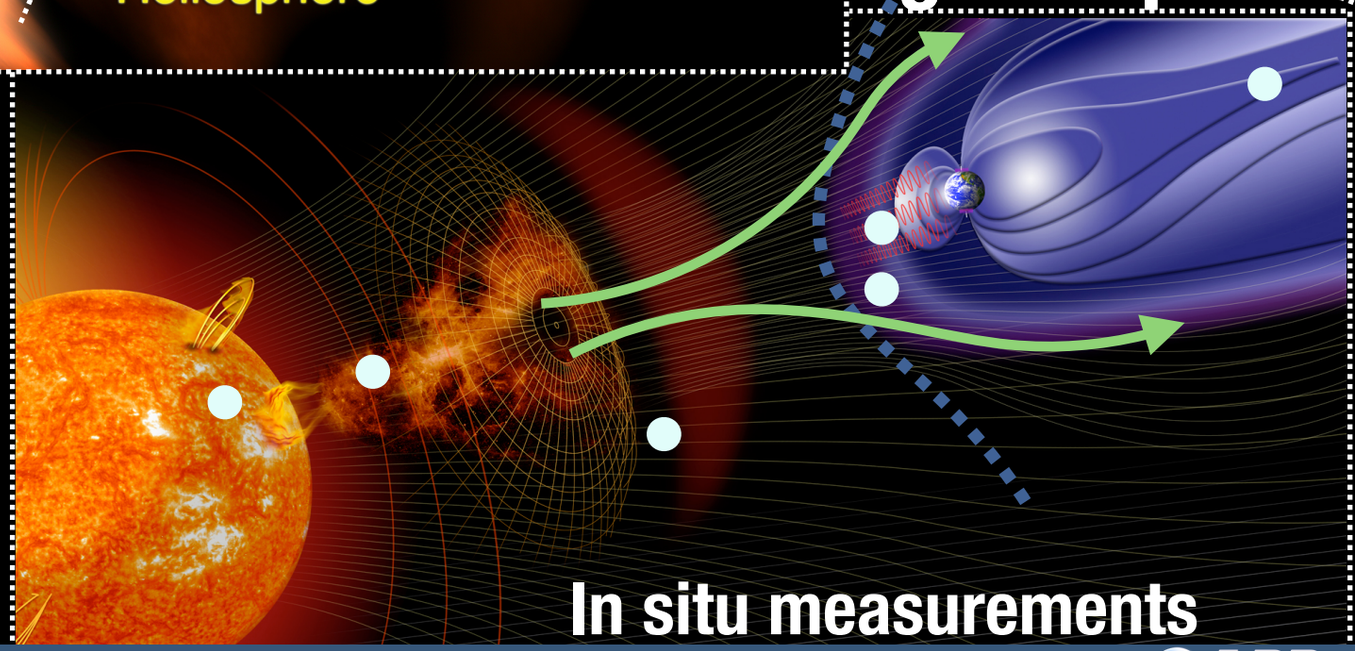
- Past, Current and Future NASA Heliophysics Missions
- Well, before current's USA administration starts their journey back to middle age….

Universality of plasma processes

- The Sun itself is a **magnetized plasma object** interacting with the interstellar plasma and magnetic field
- The **Heliosphere** is the Sun's magnetosphere.
- **Much less accessible** to in situ measurements: **analogies to the Sun-Earth** system very useful



Our magnetosphere



In situ measurements

Universality of plasma processes

Astrospheres?

Heliosphere

?

ISM

Bow Shock

Heliosheath

Voyager 1

?

Termination Shock

Voyager 2

Heliopause

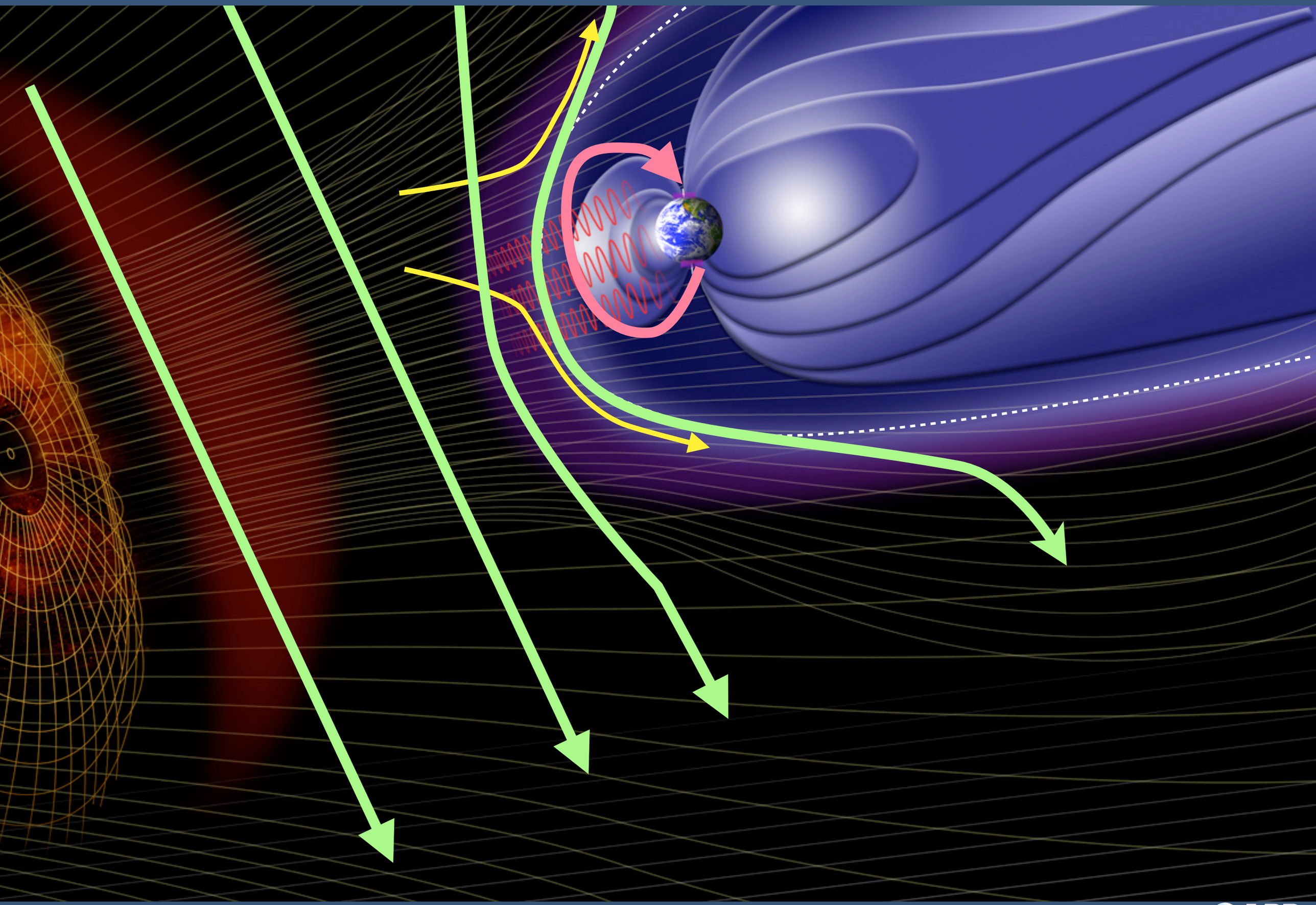
Heliosphere

Our magnetosphere

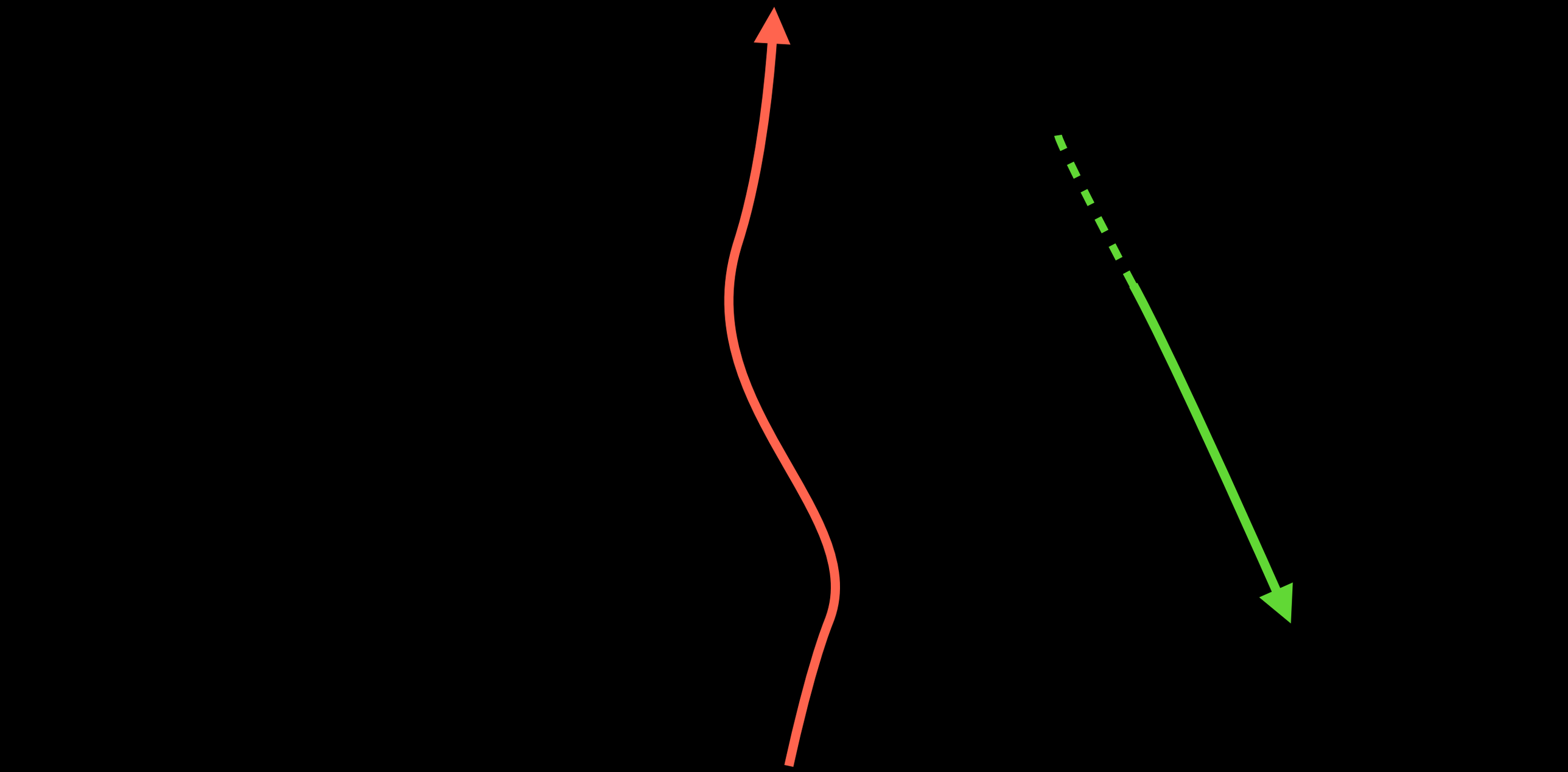
- **Magnetized plasmas are ubiquitous**
- Studying local processes give clues of what is going on up there

In situ measurements

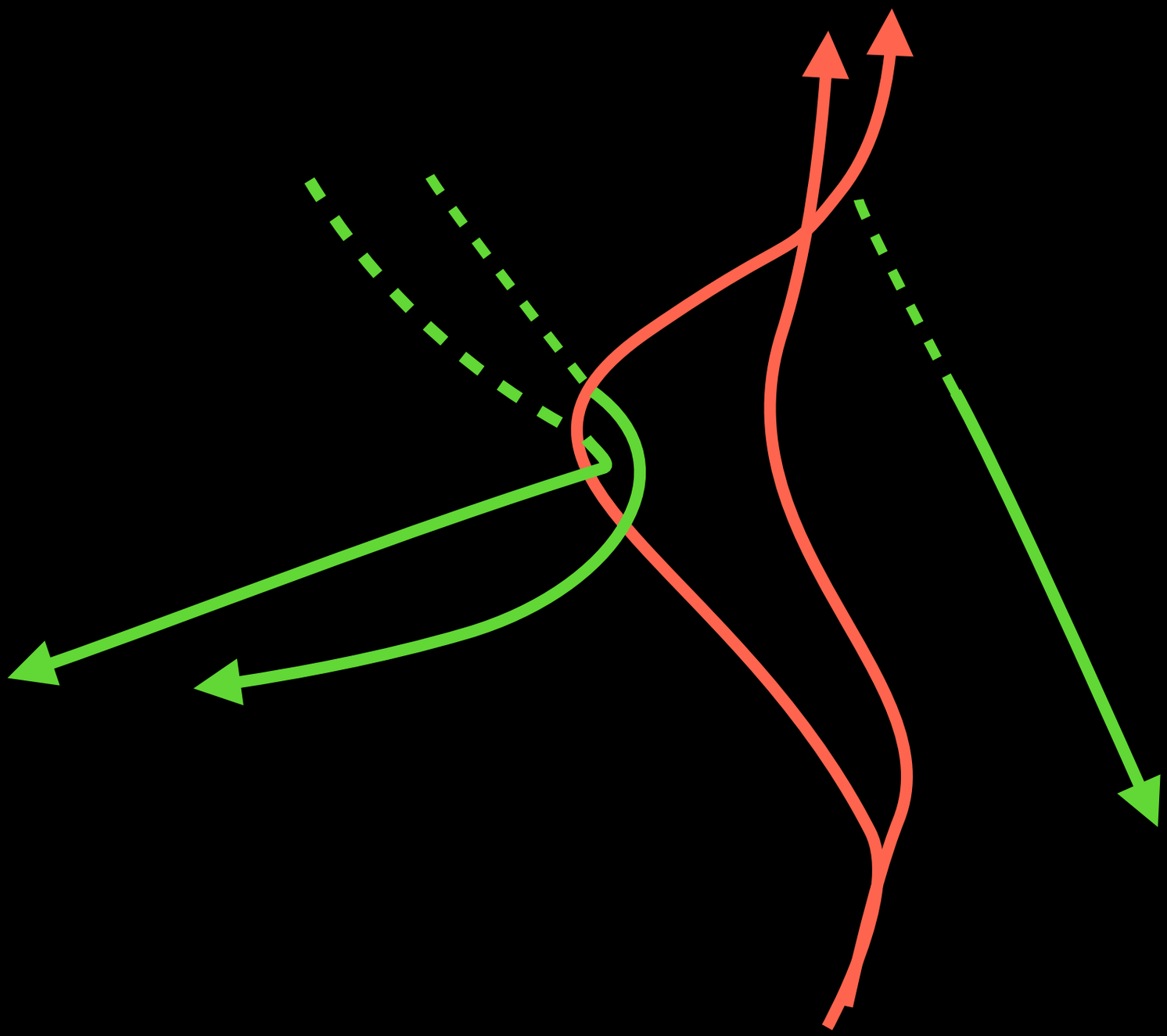
IMF interaction with the Magnetosphere



Magnetic Reconnection 101

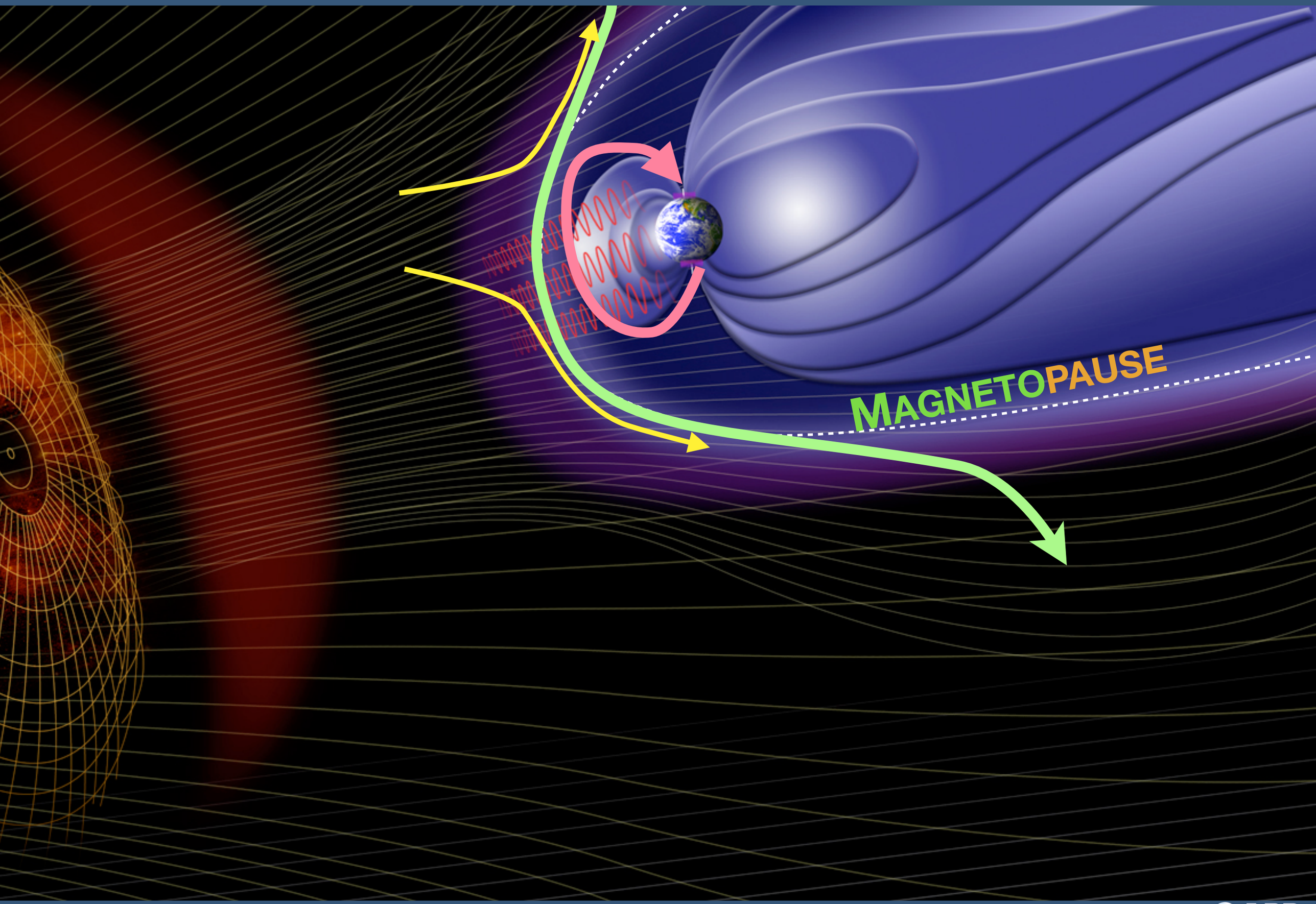


Conservation of magnetic field line connectivity

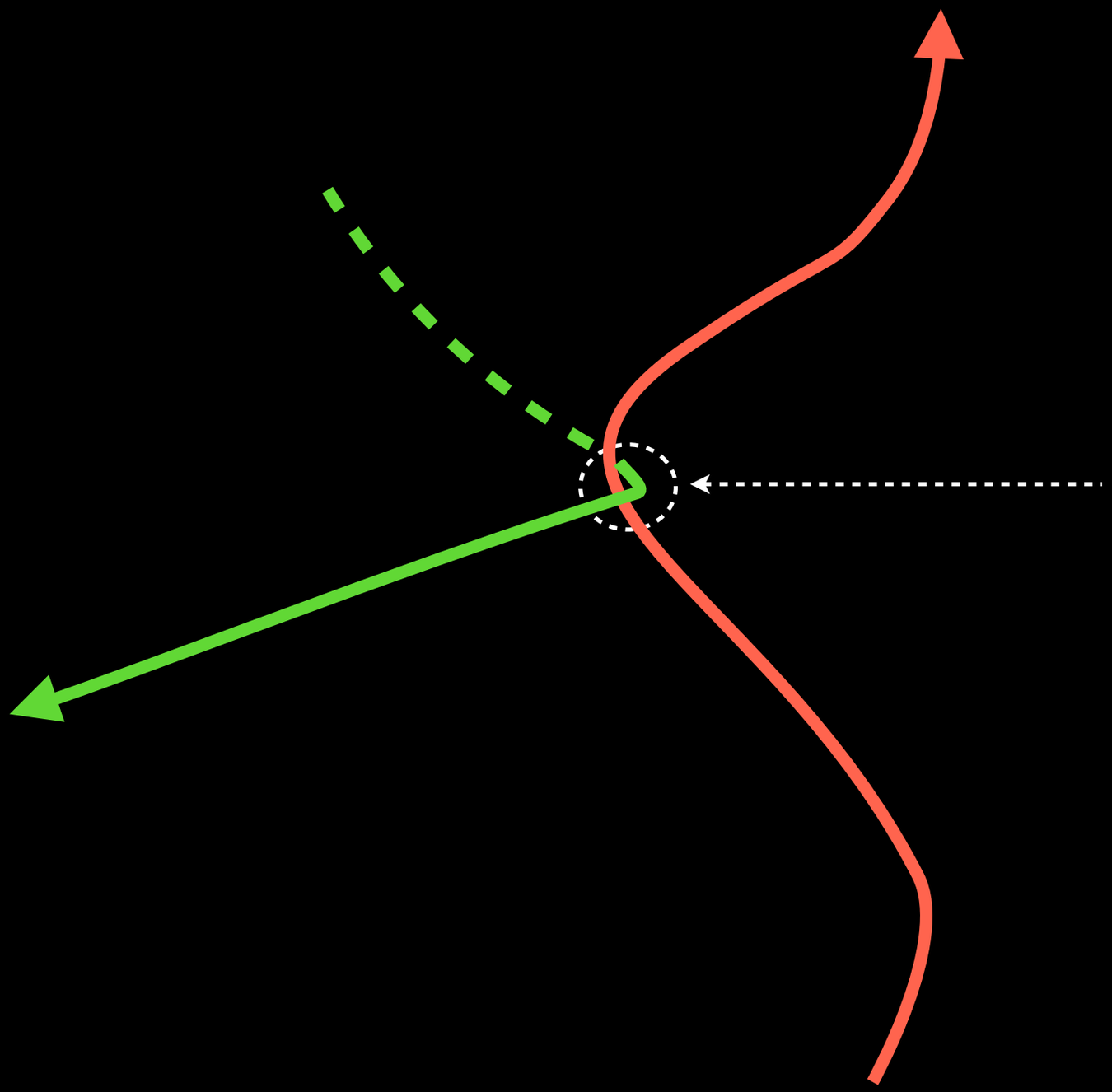


- Pulling the green line against the red one will distort them
- But they will keep their connectivity

The IMF and solar wind flow around the magnetosphere



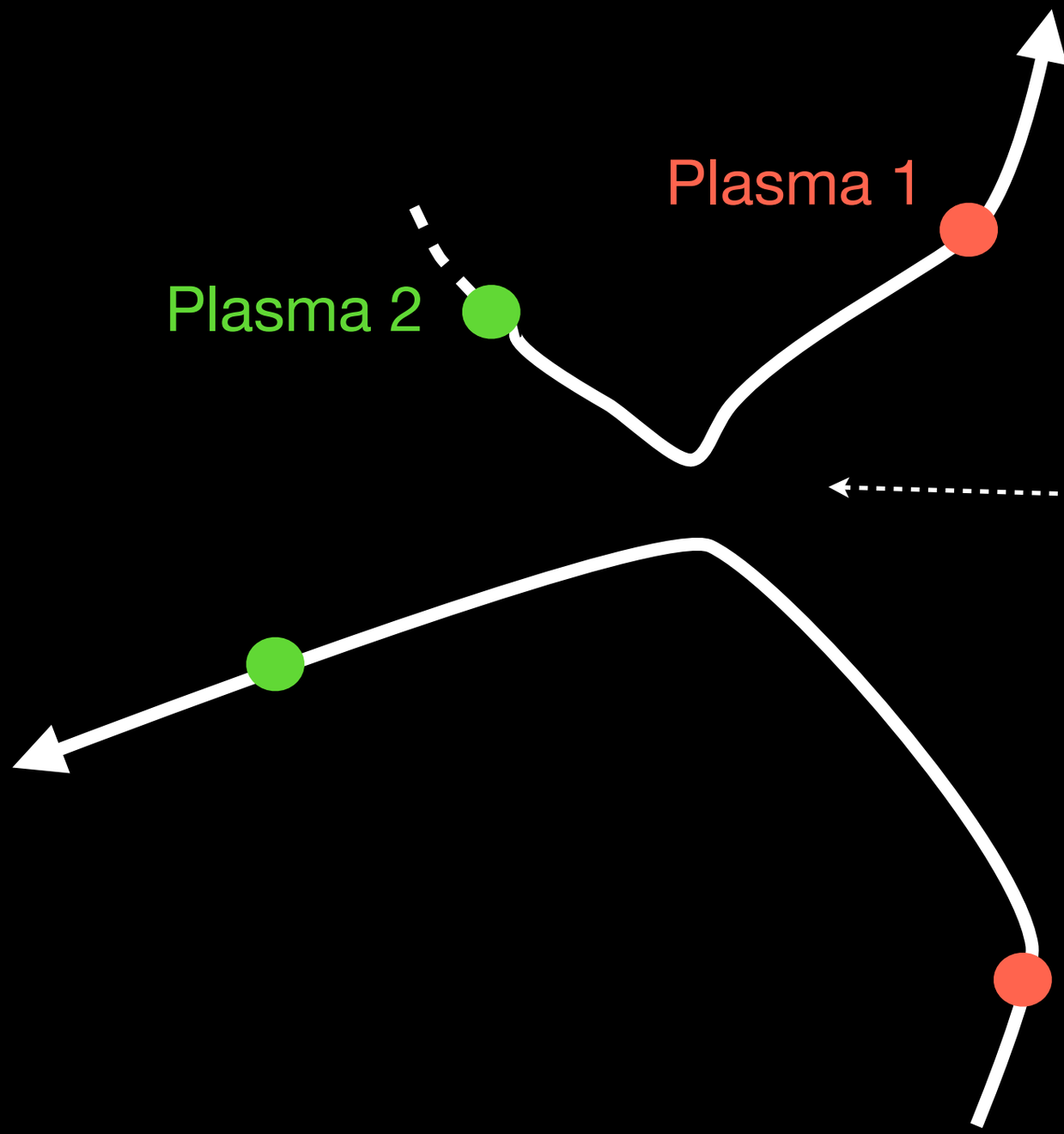
Reconnection is a non-ideal plasma process



strong gradient = non ideal processes

(we'll come back on them later)

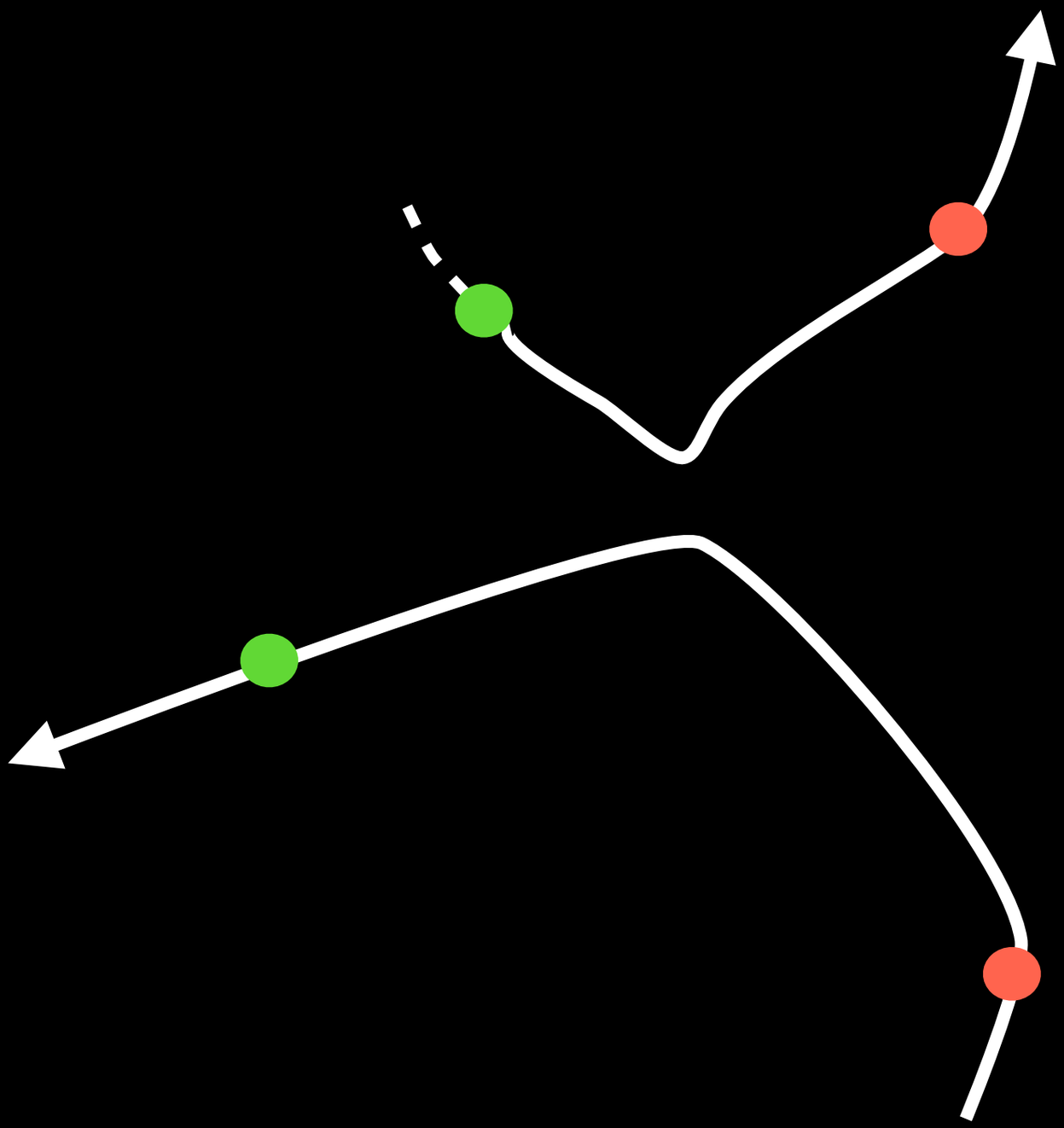
Reconnection is a non-ideal plasma process



field line
connectivity broken

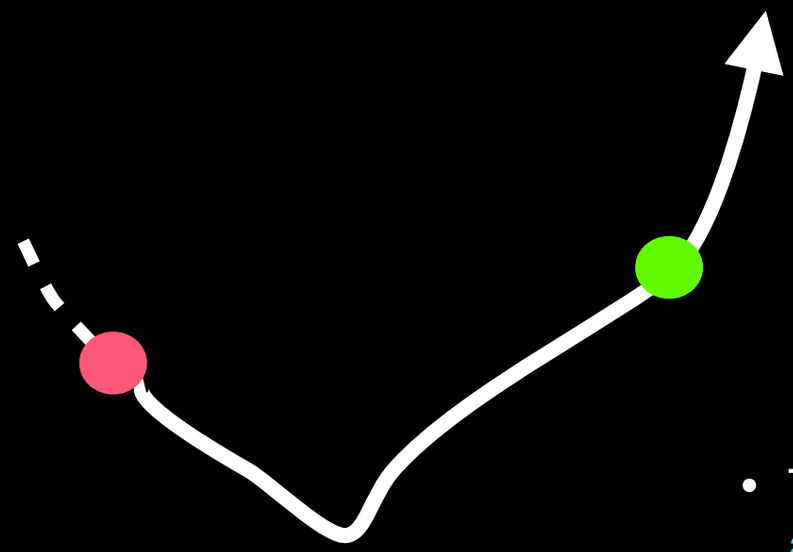
Magnetic
Re-connection

Two important consequences

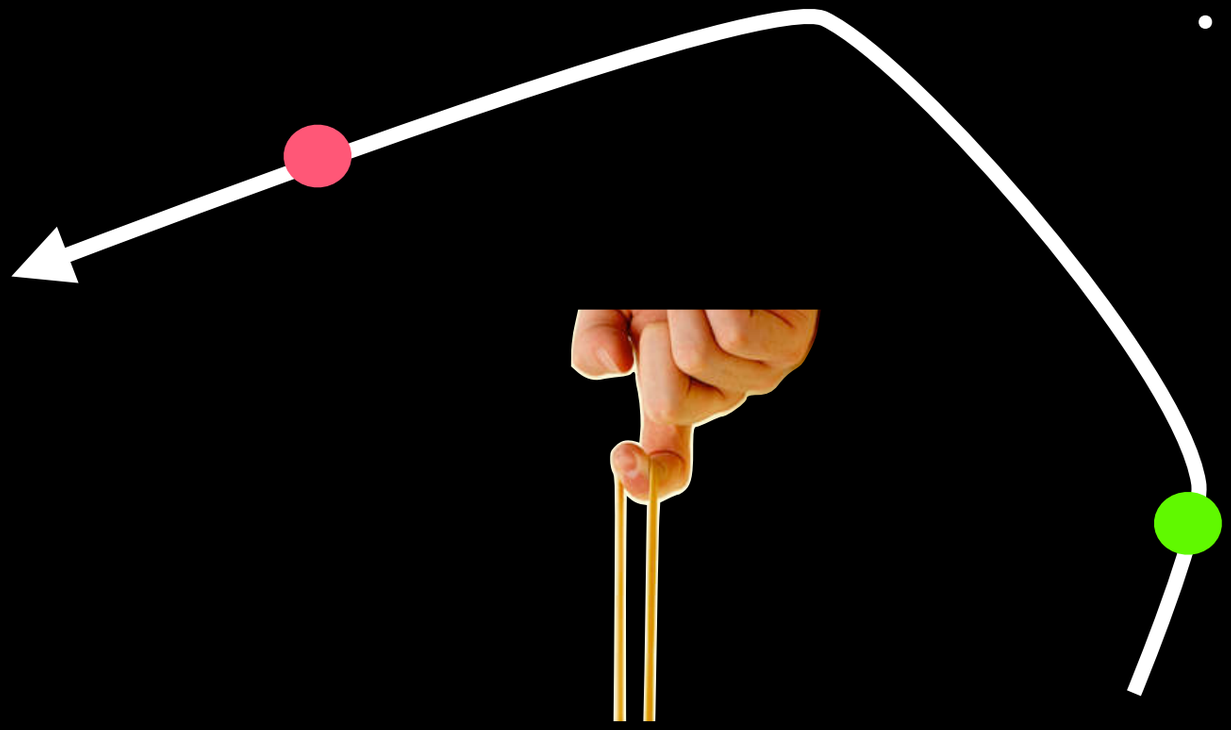


- **New connectivity** means particles can now go in regions previously inaccessible
- **Localized microphysics** processes can **change the macroscale transport** of plasma

Energy release

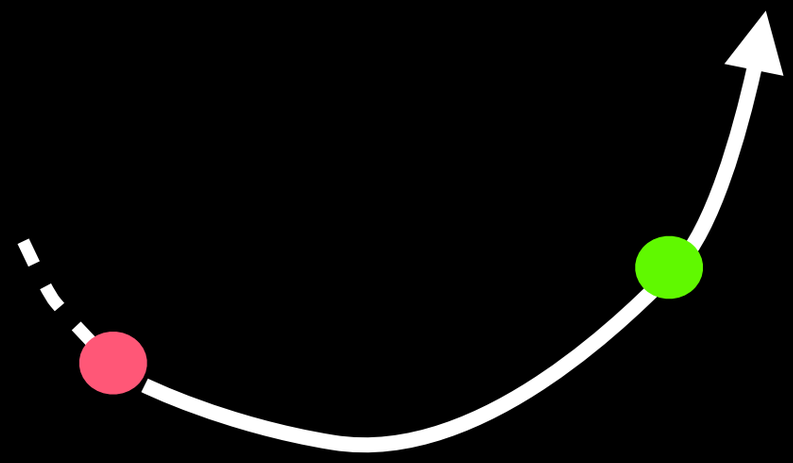


- The previous « **ideal** » **evolution led to energy accumulation** in the system

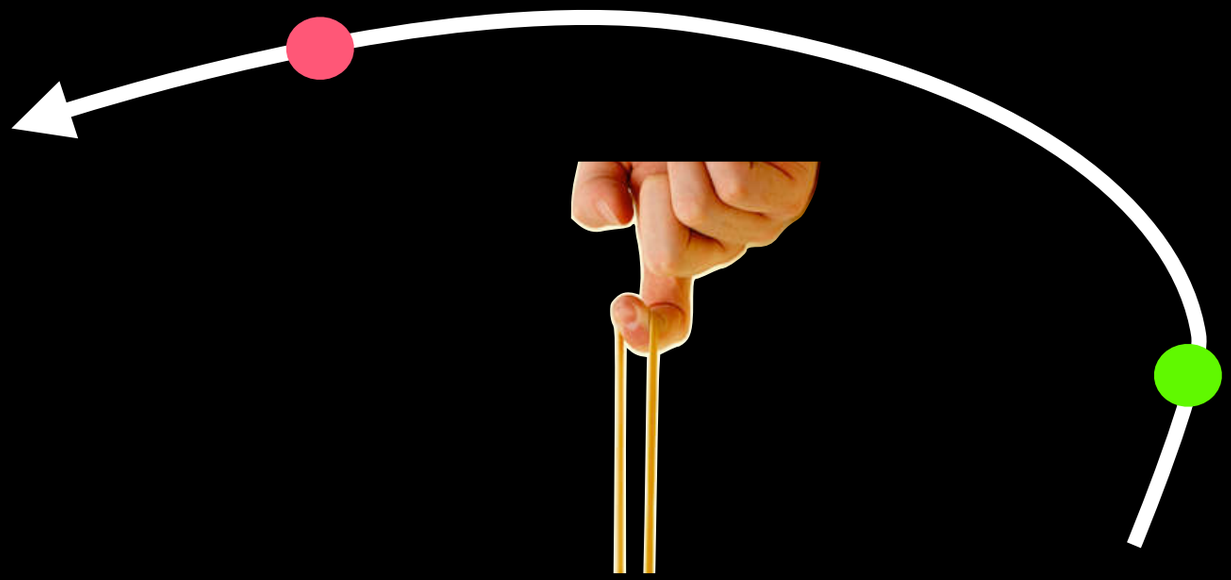


- The **sudden change of topology** allows for a rapid **release of that energy**

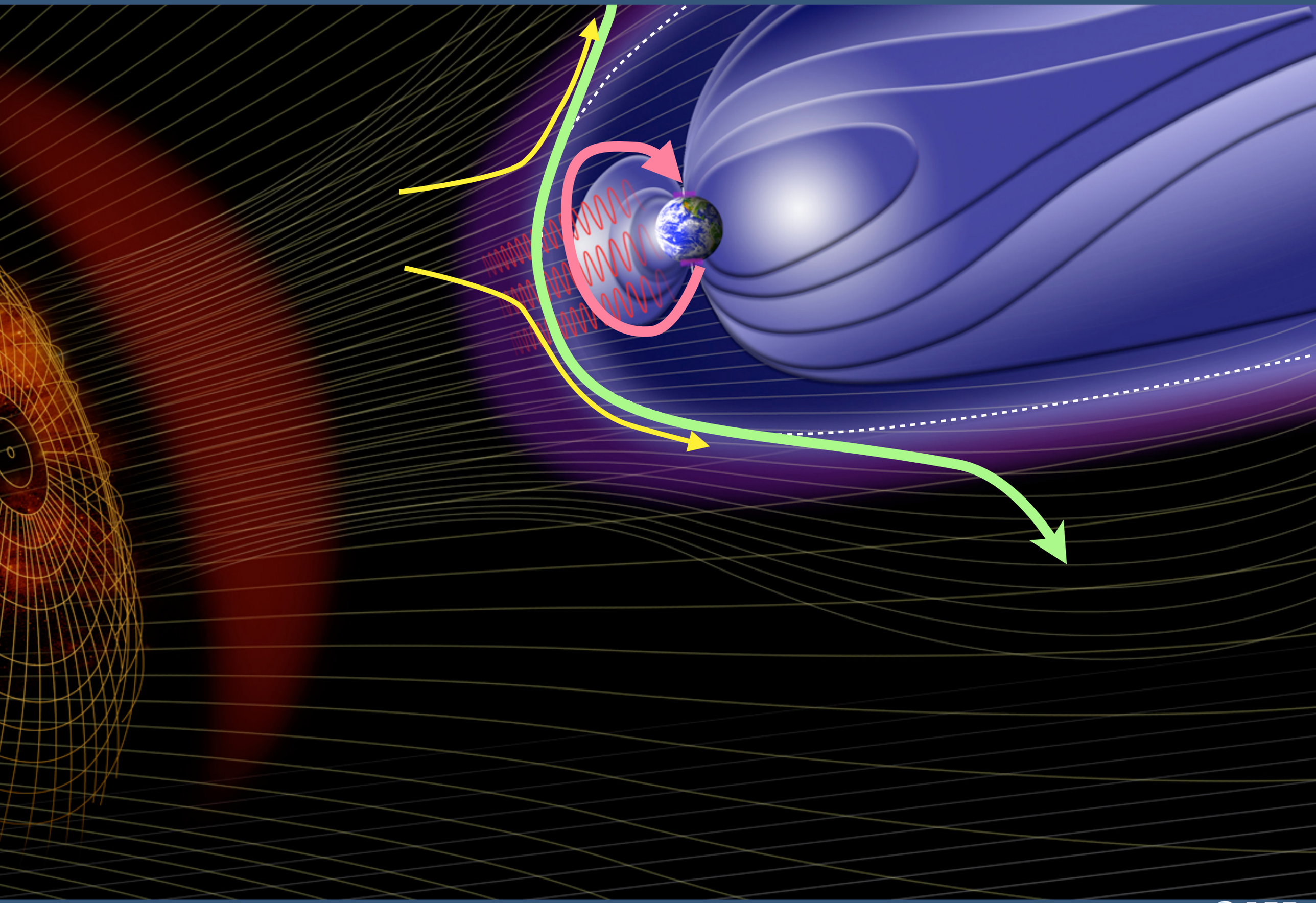
Heating and acceleration of the plasma



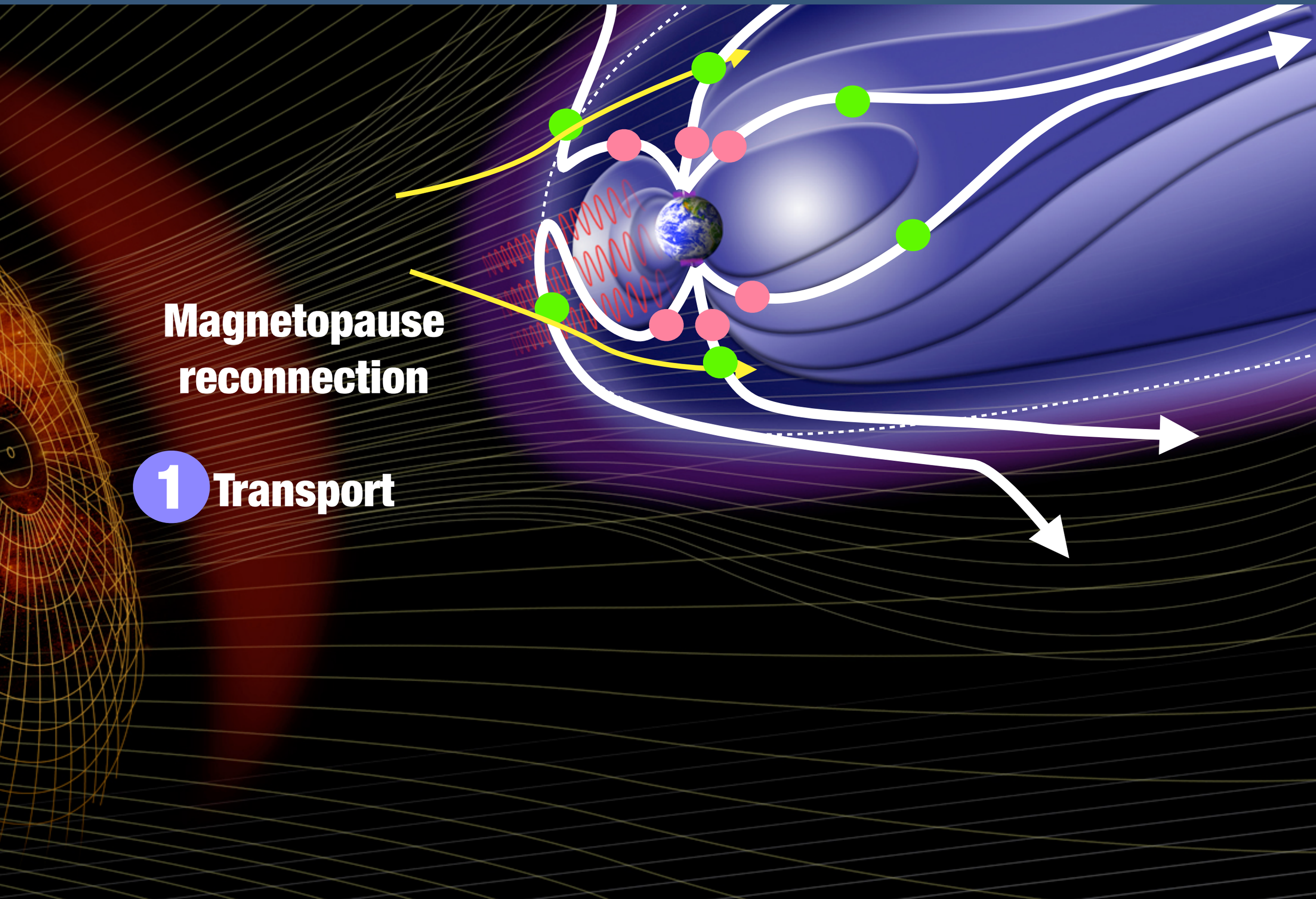
- Particles are accelerated by magnetic « tension » kind of like Perls on a rubber band
- Reconnection accelerates particles
- **Magnetic energy is transferred to kinetic and thermal plasma energy**



Reconnection at the magnetopause



Reconnection at the magnetopause

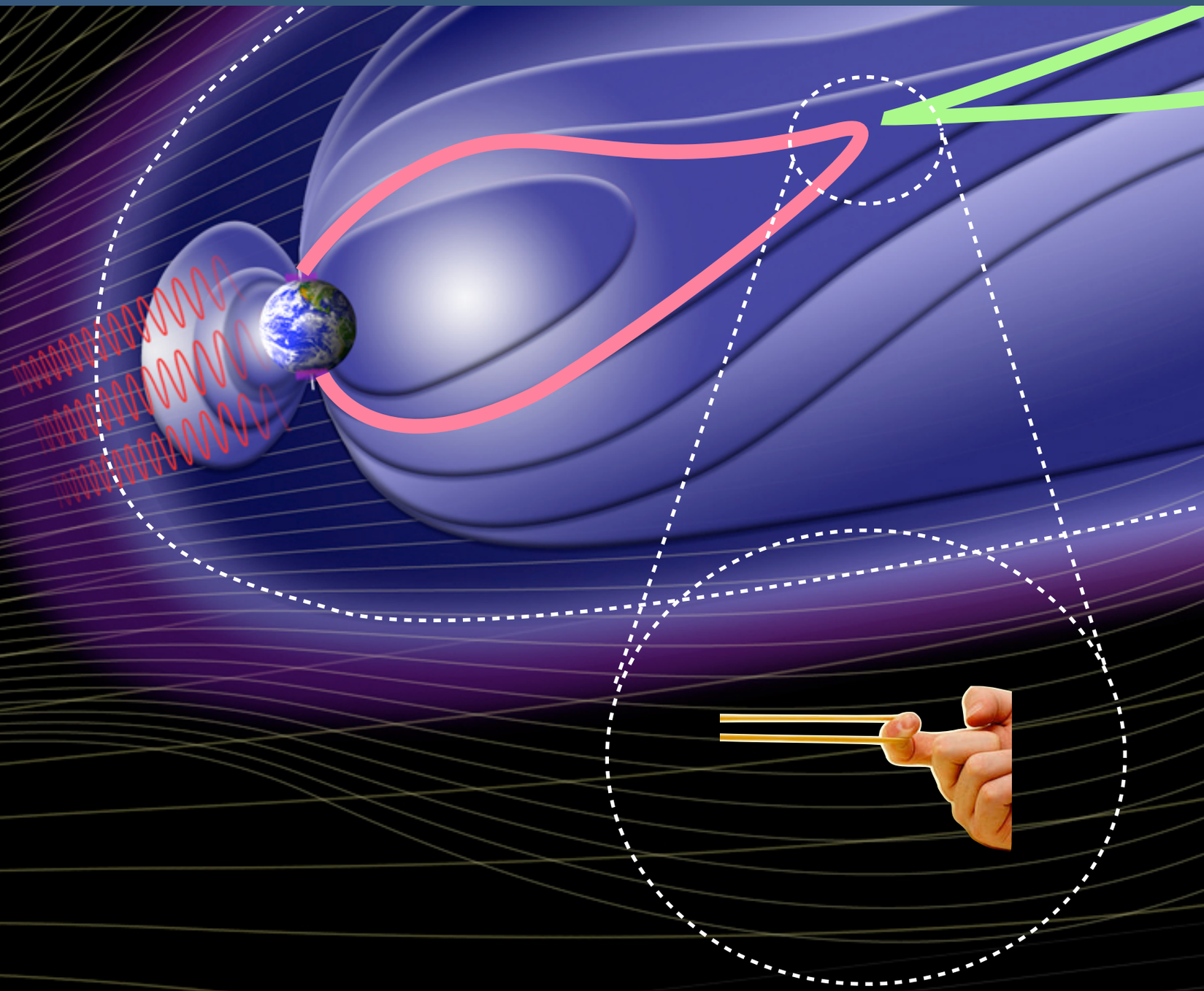


Reconnection in the magnetotail

Tail reconnection

Accélération





Universality of plasma processes: Magnetic Reconnection

Aurorae on other planets

Sun-Earth coupling

Comets' tail

Aurorae

Crab Nebula

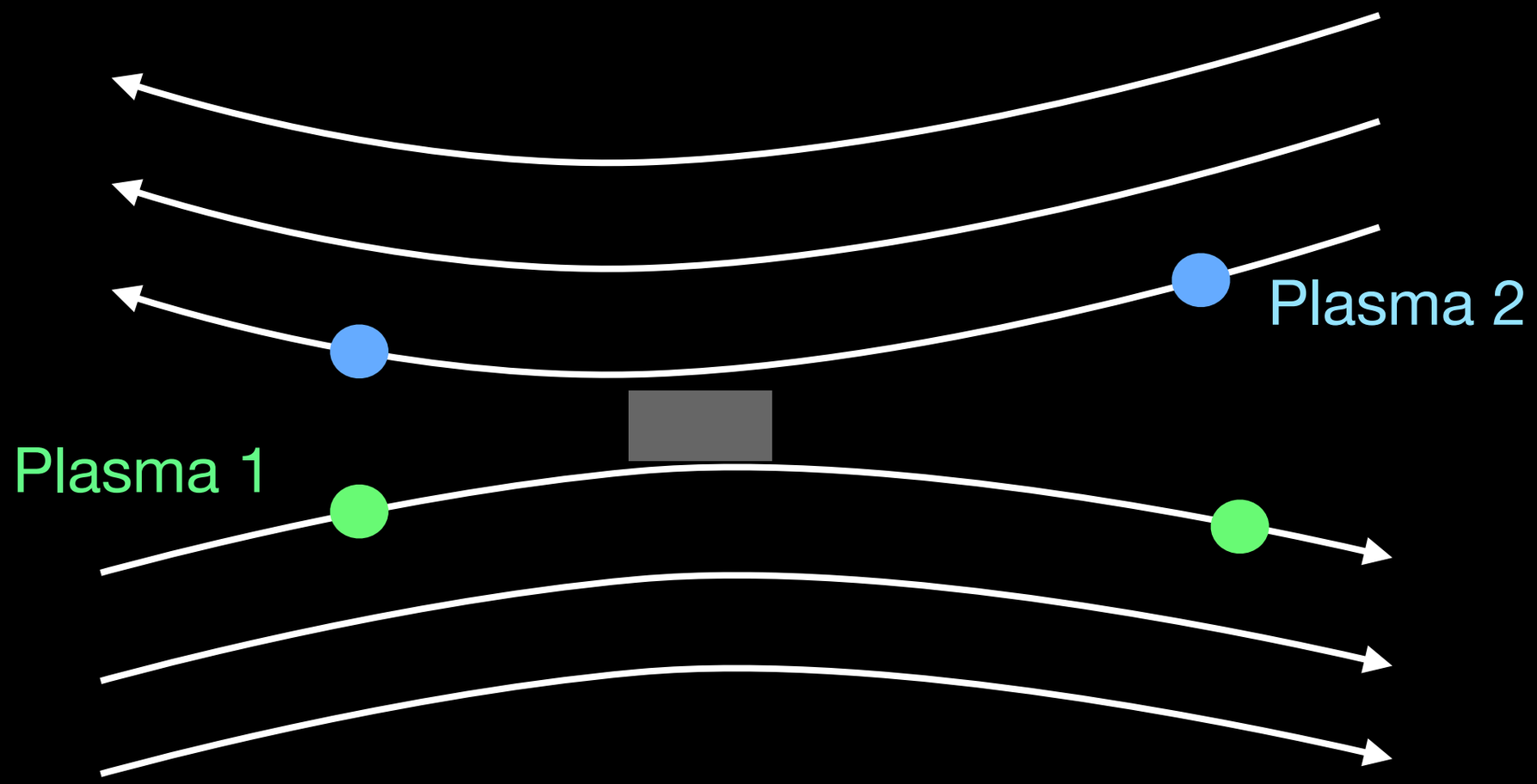
Magnetars

Sag A* Galactic

Explosions on the sun

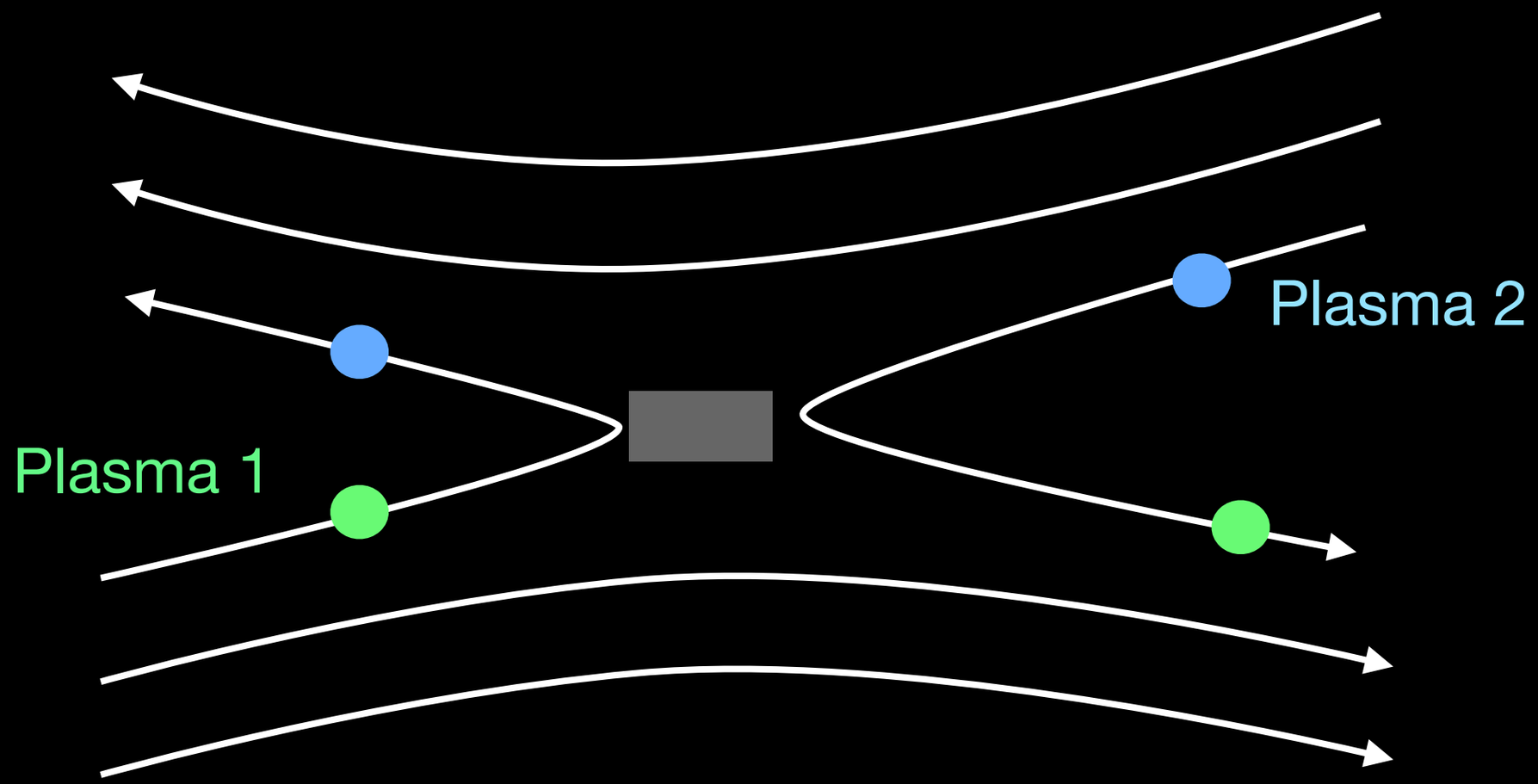
How does reconnection work?

Two magnetized plasmas in « contact »

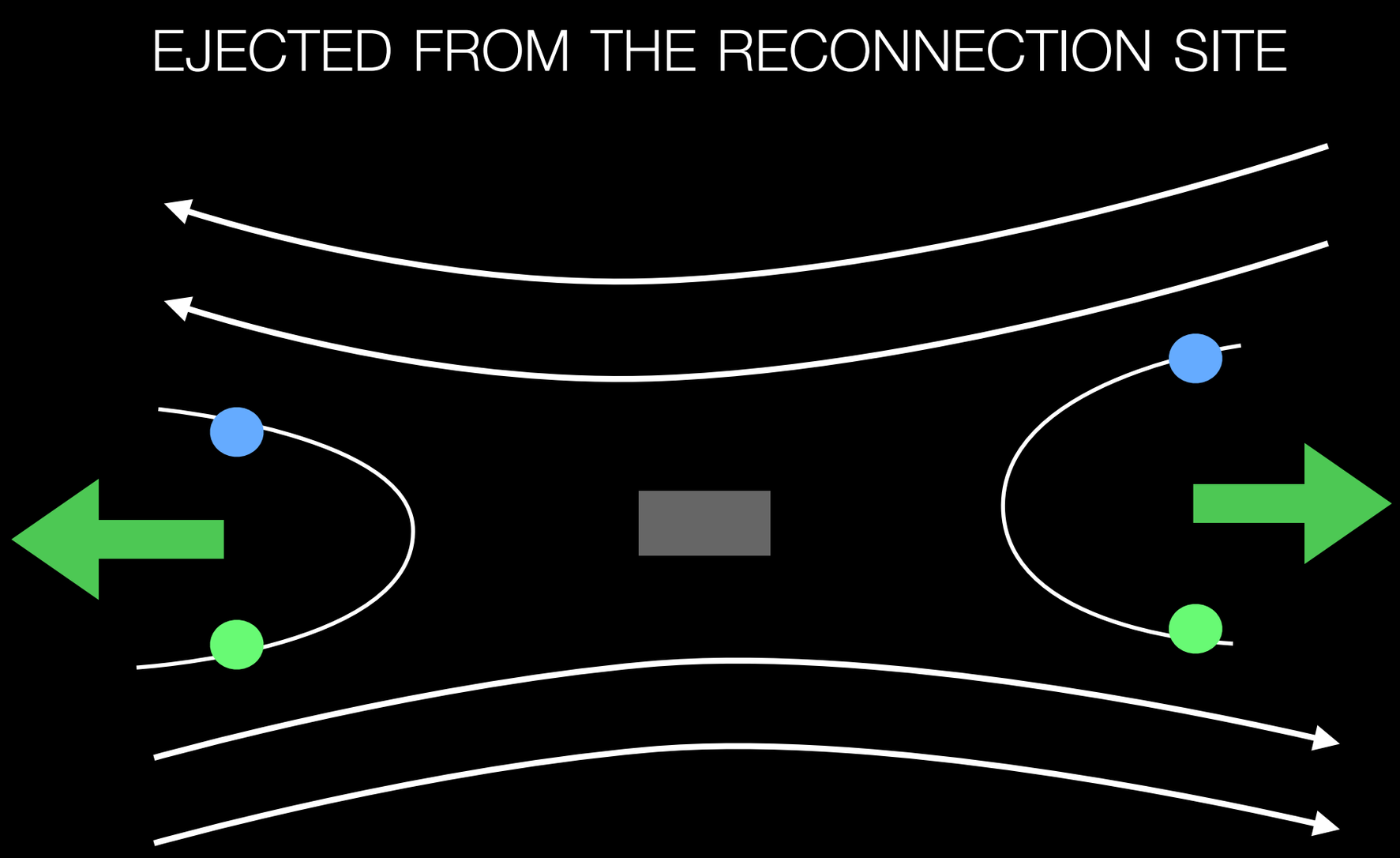


How does reconnection work?

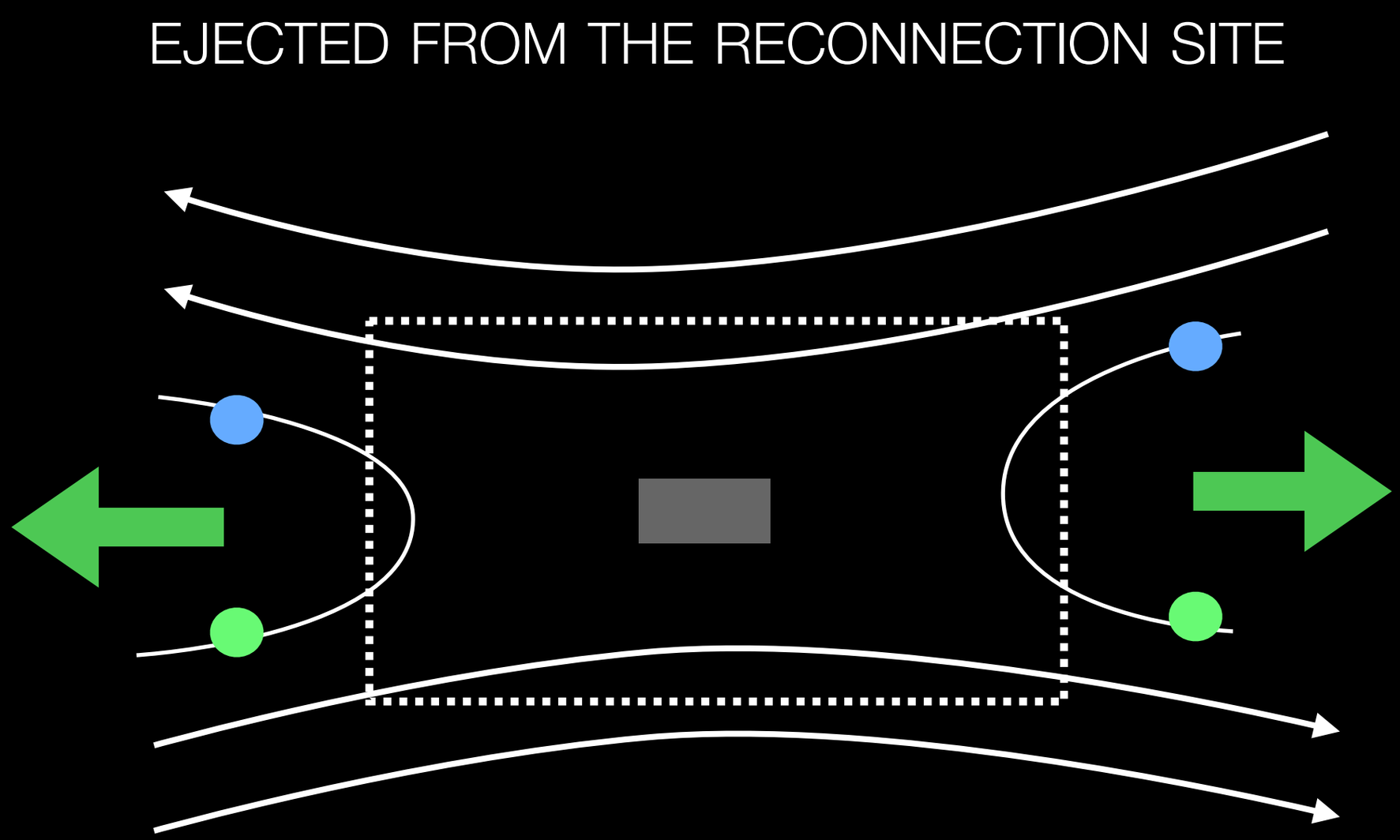
CAN BE RECONNECTED



How does reconnection work?

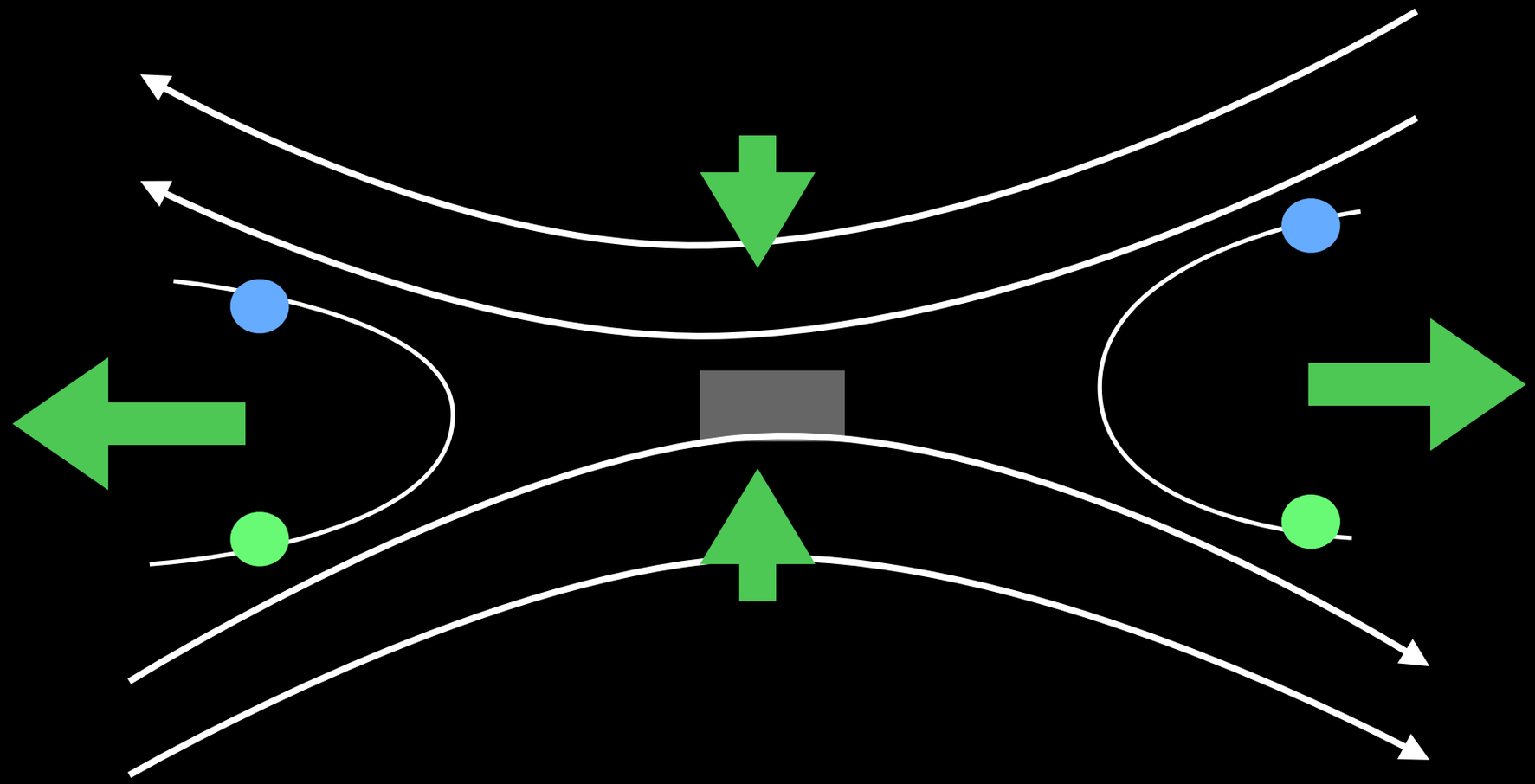


How does reconnection work?



How does reconnection work?

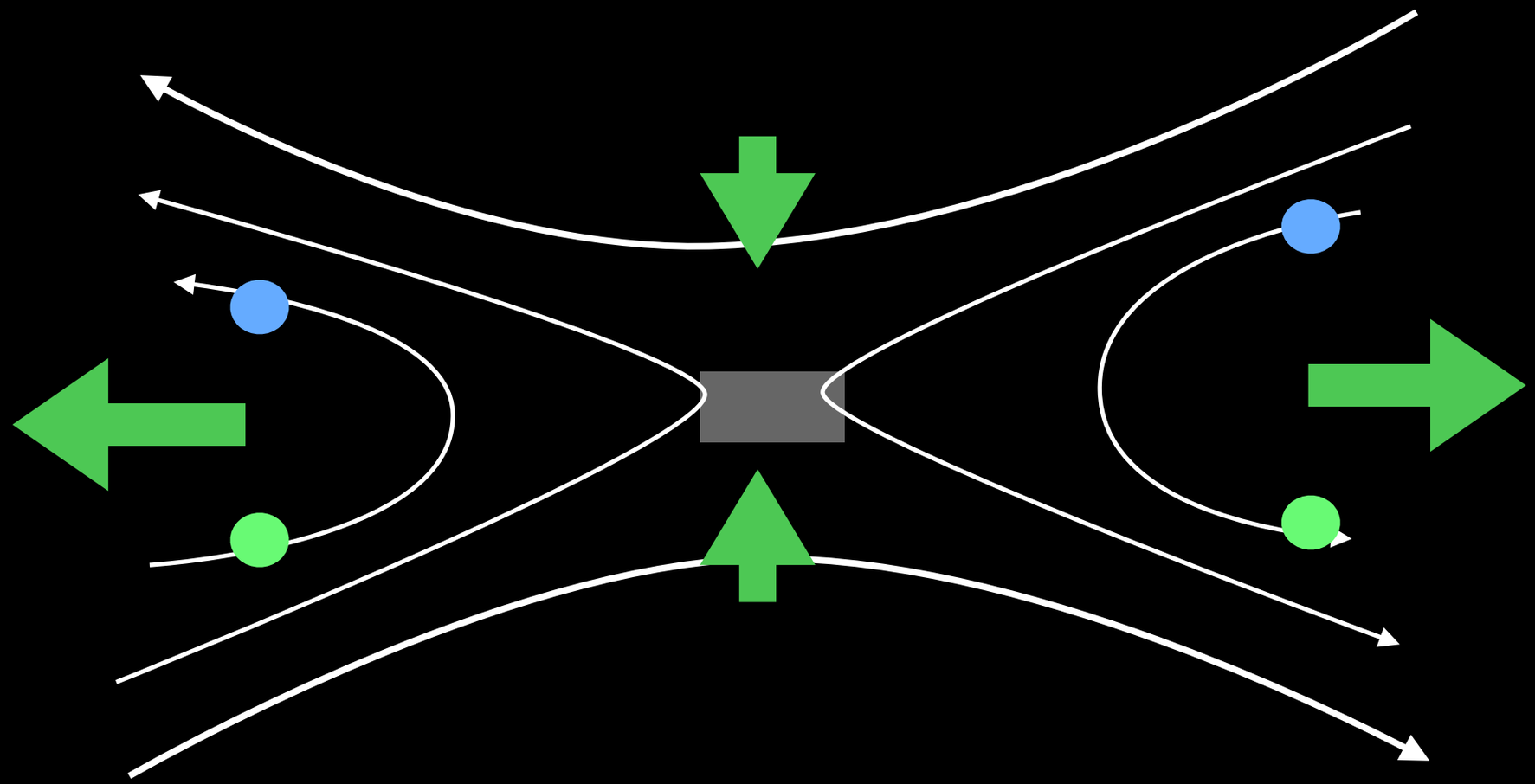
This drives the pulling of upstream flux and plasma



How does reconnection work?

WHICH IS RECONNECTED AND EJECTED

ETC. ETC. AND THE PROCESS IS SELF MAINTAINED



MagnetoHydroDynamics

- **Hydrodynamics**: only consider the evolution « macroscopic » quantities
  - Density, momentum, pressure, energy…
- **Magneto**: fluid is conducting electrical current and respond to magnetic forces

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{v}) = 0$$

- Mass conservation

$$\rho \left(\frac{\partial \mathbf{v}}{\partial t} + \mathbf{v} \cdot \nabla \mathbf{v} \right) = -\nabla P + \mathbf{J} \times \mathbf{B}$$

- Momentum balance

$$\frac{\partial \mathbf{B}}{\partial t} = \nabla \times (\mathbf{v} \times \mathbf{B}) + \eta \nabla^2 \mathbf{B}$$

- Resistive Induction Equation

Notes :

- Resistivity is linked to electron/ion collisions, leading to thermalization and dissipation
- Only two spatial scales here: system scale and resistive dissipation scale

MagnetoHydroDynamics

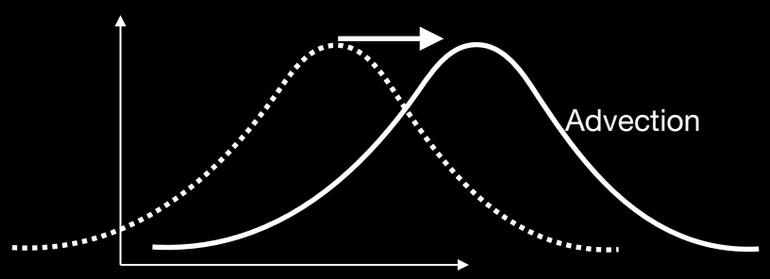
$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{v}) = 0$$

$$\rho \left(\frac{\partial \mathbf{v}}{\partial t} + \mathbf{v} \cdot \nabla \mathbf{v} \right) = -\nabla P + \mathbf{J} \times \mathbf{B}$$

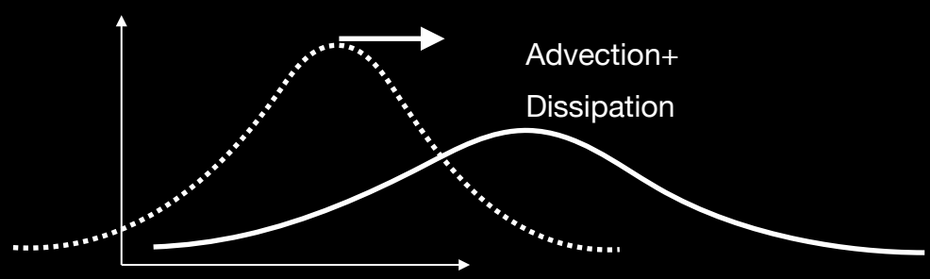
$$\frac{\partial \mathbf{B}}{\partial t} = \nabla \times (\mathbf{v} \times \mathbf{B}) + \eta \nabla^2 \mathbf{B}$$

- Mass conservation
- Momentum balance
- Resistive Induction Equation

‘Induction equation’ $\frac{\partial \mathbf{B}}{\partial t} = \nabla \times (\mathbf{v} \times \mathbf{B}) + \eta \nabla^2 \mathbf{B}$

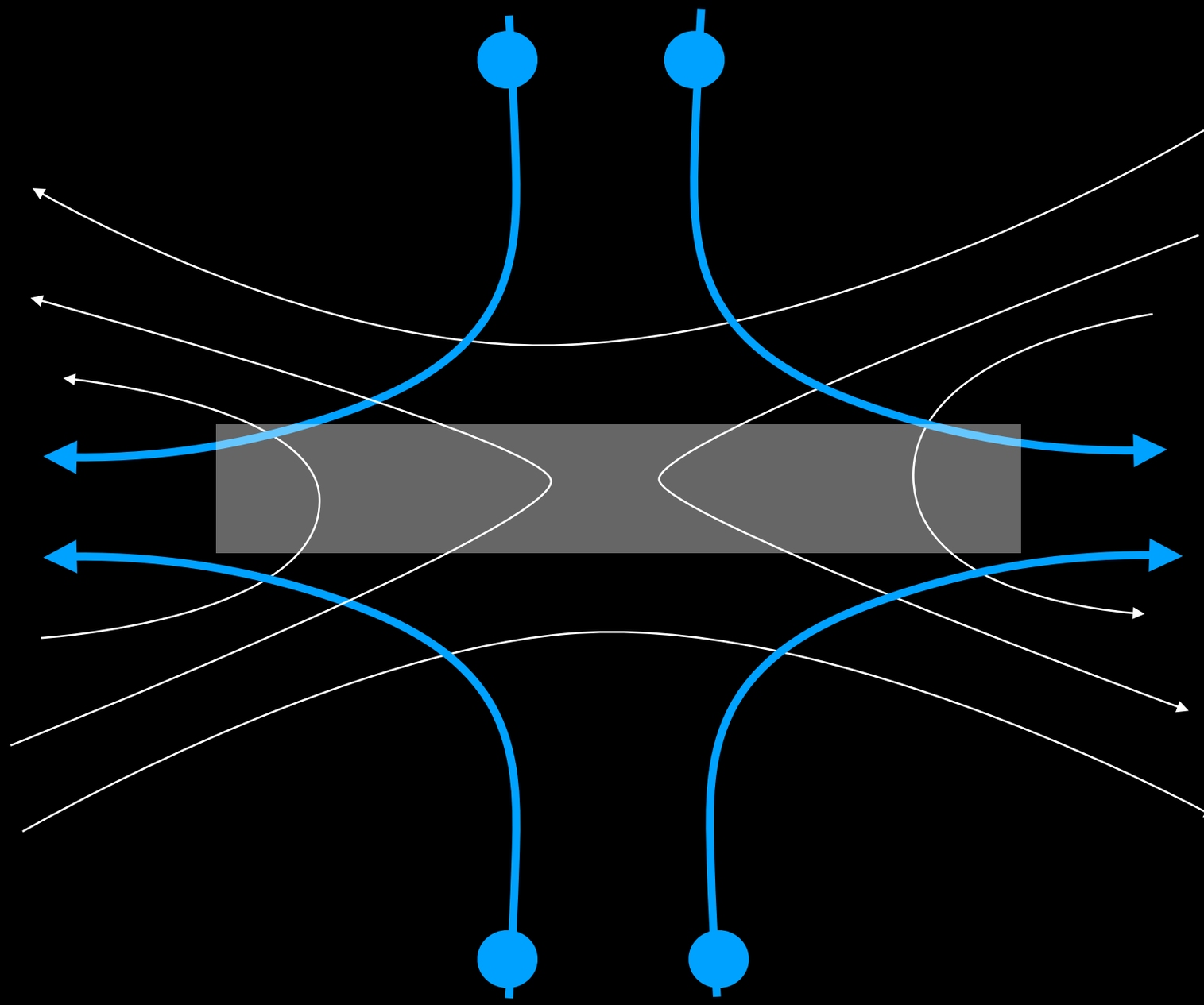


Magnetic Reynolds Number $\mathcal{R} = \frac{VL}{\eta}$



- Resistivity is linked to electron/ion collisions, leading to thermalization and dissipation
- Only two spatial scales here: system scale and resistive dissipation scale

FLOW IN MAGNETIC RECONNECTION



$$\frac{\partial \mathbf{B}}{\partial t} = \nabla \times (\mathbf{v} \times \mathbf{B})$$

$$\frac{\partial \mathbf{B}}{\partial t} = \eta \nabla^2 \mathbf{B}$$

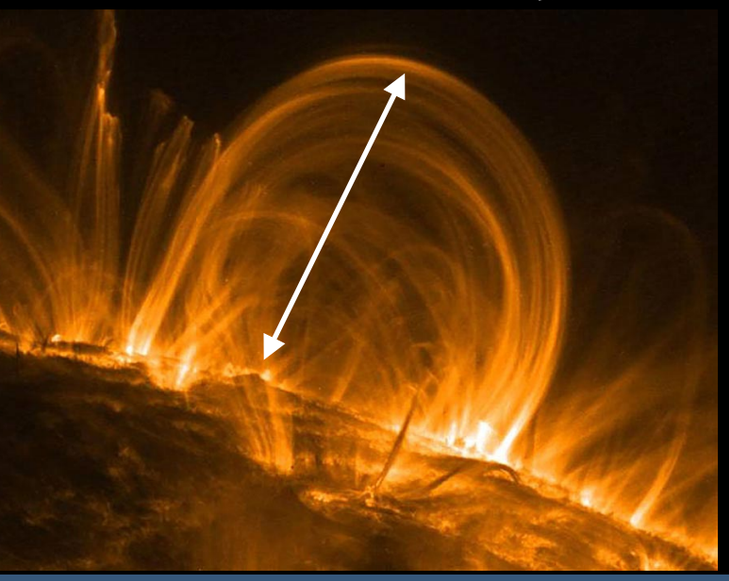
Nearly collisionless resistive reconnection: too slow to be true

$$\frac{\partial \mathbf{B}}{\partial t} = \nabla \times (\mathbf{v} \times \mathbf{B})$$

tiny

$$\frac{\partial \mathbf{B}}{\partial t} = \eta \nabla^2 \mathbf{B}$$

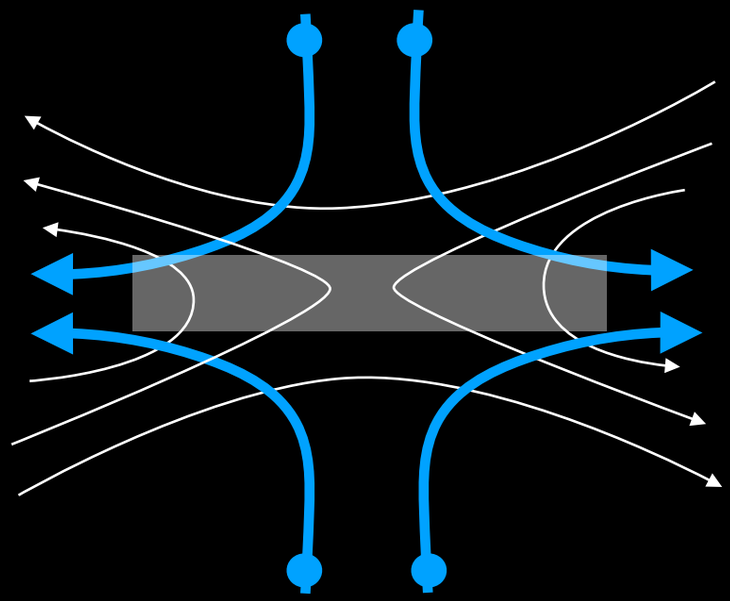
Super large



Magnetic reconnection is much too slow in this formalism: we forget some physics!

Hall MagnetoHydroDynamics

Describing the evolution of the plasma as a single fluid does not hold at a certain scale…







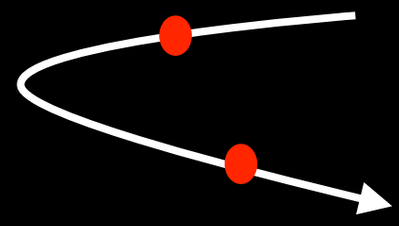
Ions are much heavier than electrons and have a much larger inertial length scale

HALL MAGNETOHYDRODYNAMICS

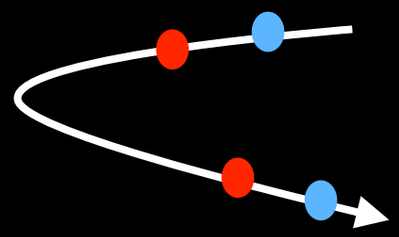
$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{v}) = 0$$

$$\rho \left(\frac{\partial \mathbf{v}}{\partial t} + \mathbf{v} \cdot \nabla \mathbf{v} \right) = -\nabla P + \mathbf{J} \times \mathbf{B}$$

$$\frac{\partial \mathbf{B}}{\partial t} = \nabla \times (\mathbf{v}_e \times \mathbf{B})$$

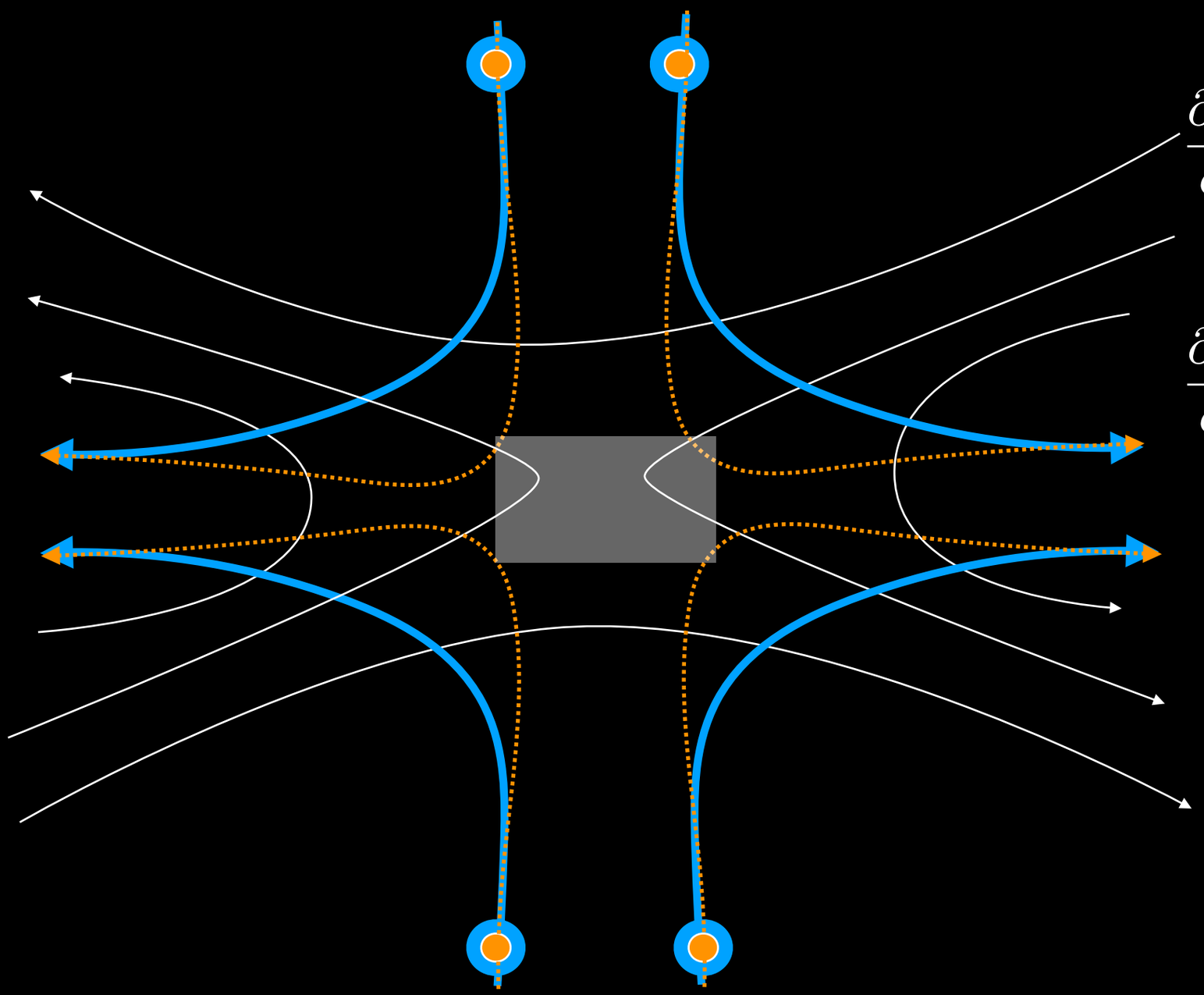


SINGLE FLUID FROZEN IN THE MAGNETIC FIELD



ONLY ELECTRONS ARE ASSUMED TO BE FROZEN IN B. ION INERTIA ALLOW THEM TO DETACH AT SMALL SCALES

HALL MAGNETOHYDRODYNAMICS



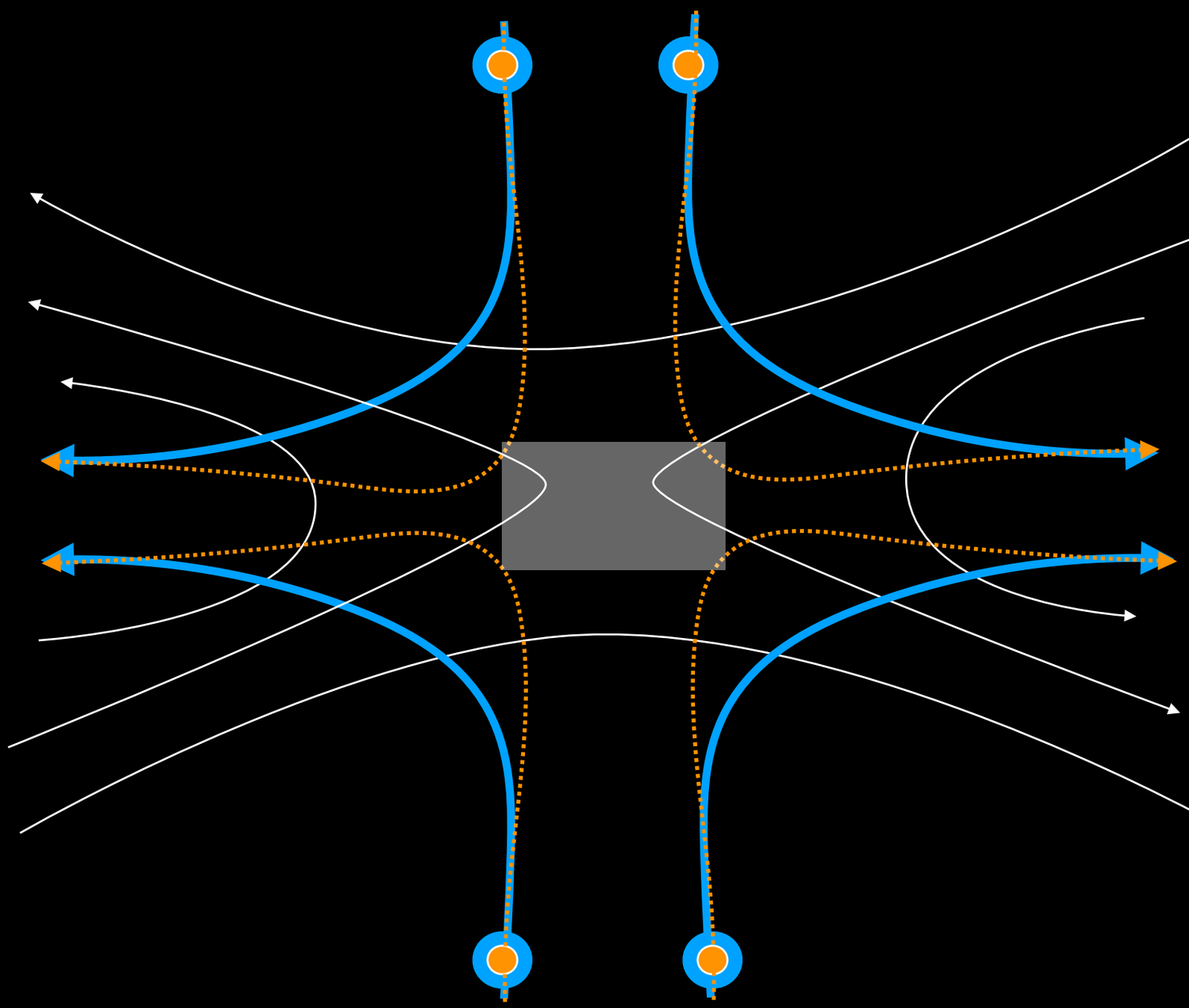
$$\frac{\partial \mathbf{B}}{\partial t} = \nabla \times (\mathbf{v} \times \mathbf{B})$$

$$\frac{\partial \mathbf{B}}{\partial t} = \nabla \times (\mathbf{v}_e \times \mathbf{B})$$

$$\frac{\partial \mathbf{B}}{\partial t} = \eta \nabla^2 \mathbf{B}$$

Is it enough?

HALL MAGNETOHYDRODYNAMICS



$$\frac{\partial \mathbf{B}}{\partial t} = \nabla \times (\mathbf{v} \times \mathbf{B})$$

$$\frac{\partial \mathbf{B}}{\partial t} = \nabla \times (\mathbf{v}_e \times \mathbf{B})$$

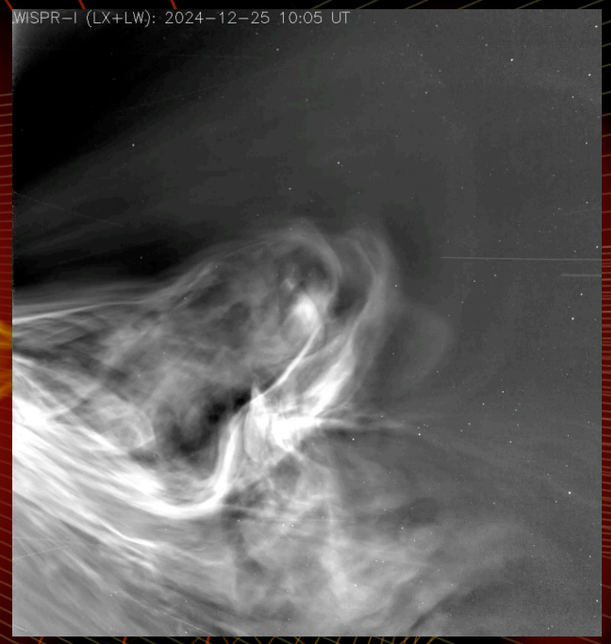
$$\frac{\partial \mathbf{B}}{\partial t} = \eta \nabla^2 \mathbf{B}$$

Fluid: macroscopic quantities are defined **locally**

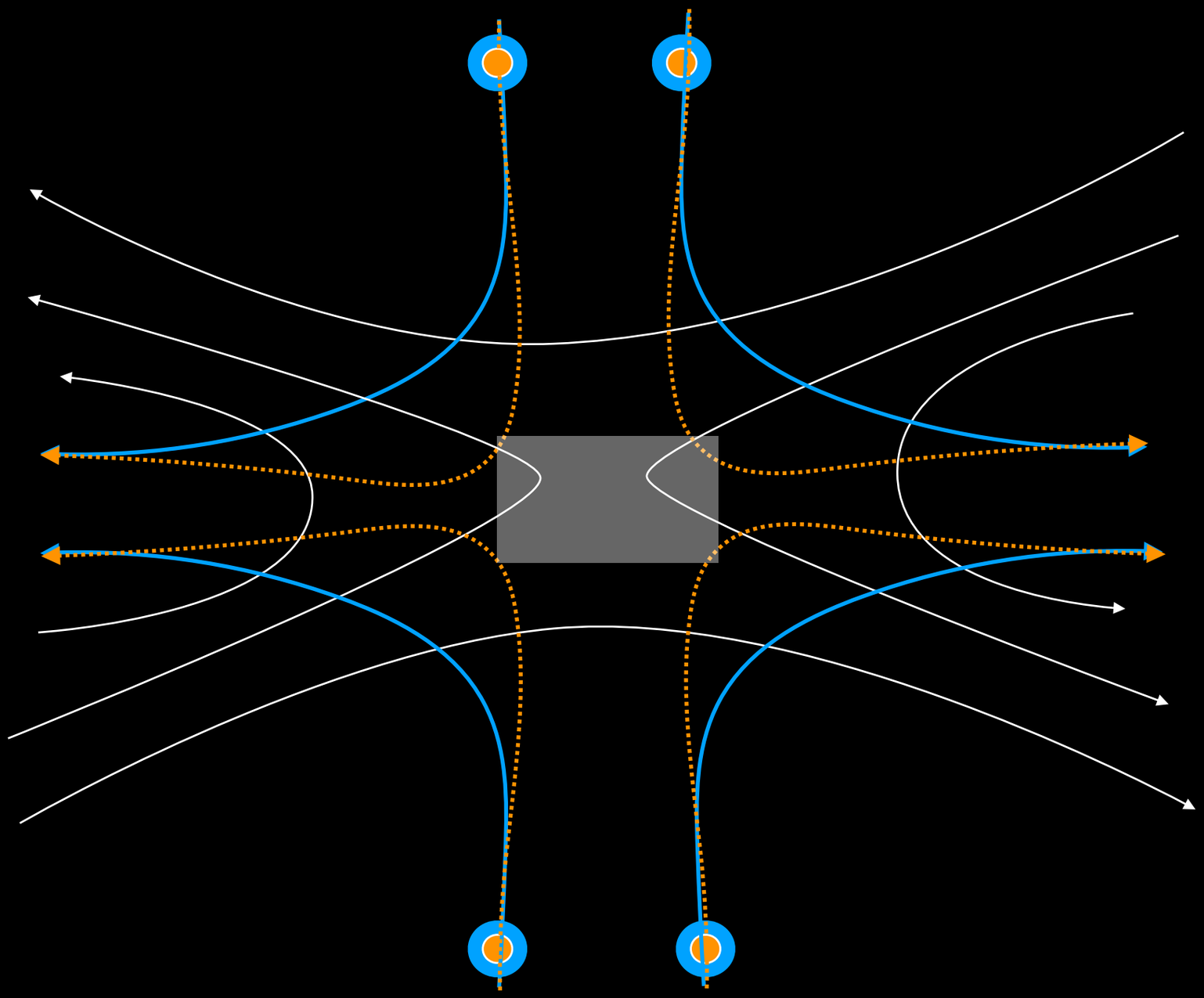
Not always true… actually almost always wrong!

Space Plasmas are Collisionless

~Particle mean free path

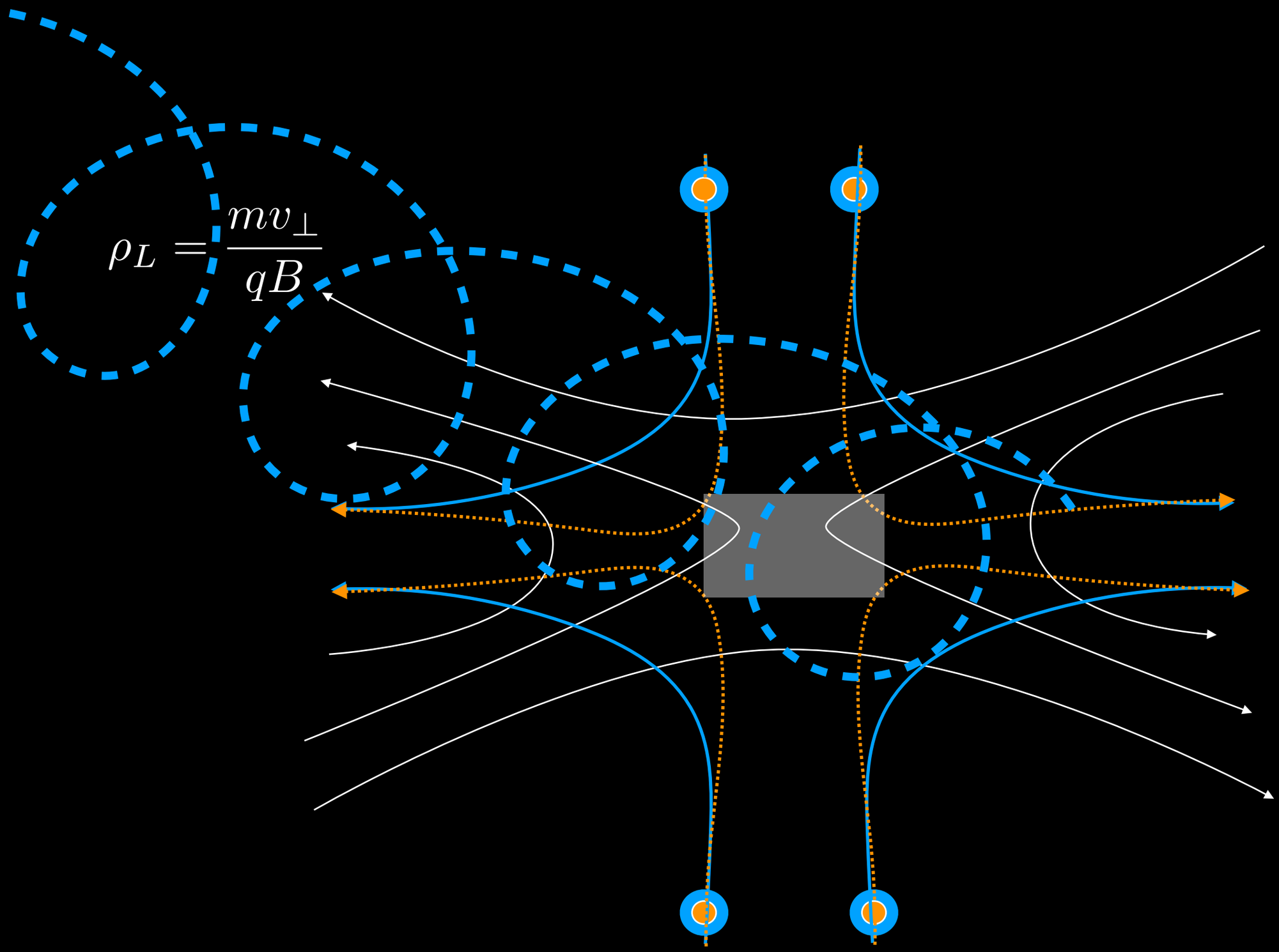


Variation Scales can be of the Order of Particle Larmor Scale



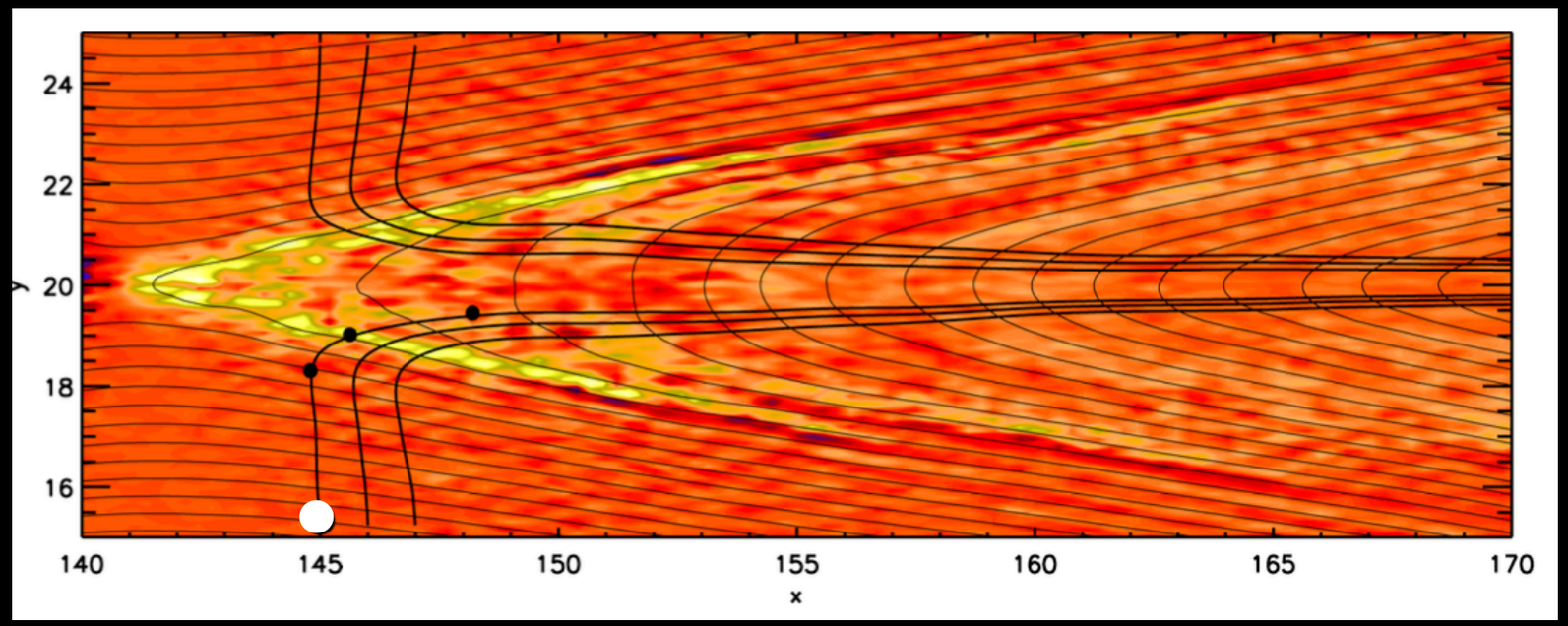
The bulk flow is « just » the local average of the particle velocities
Particles do **not** follow the same path

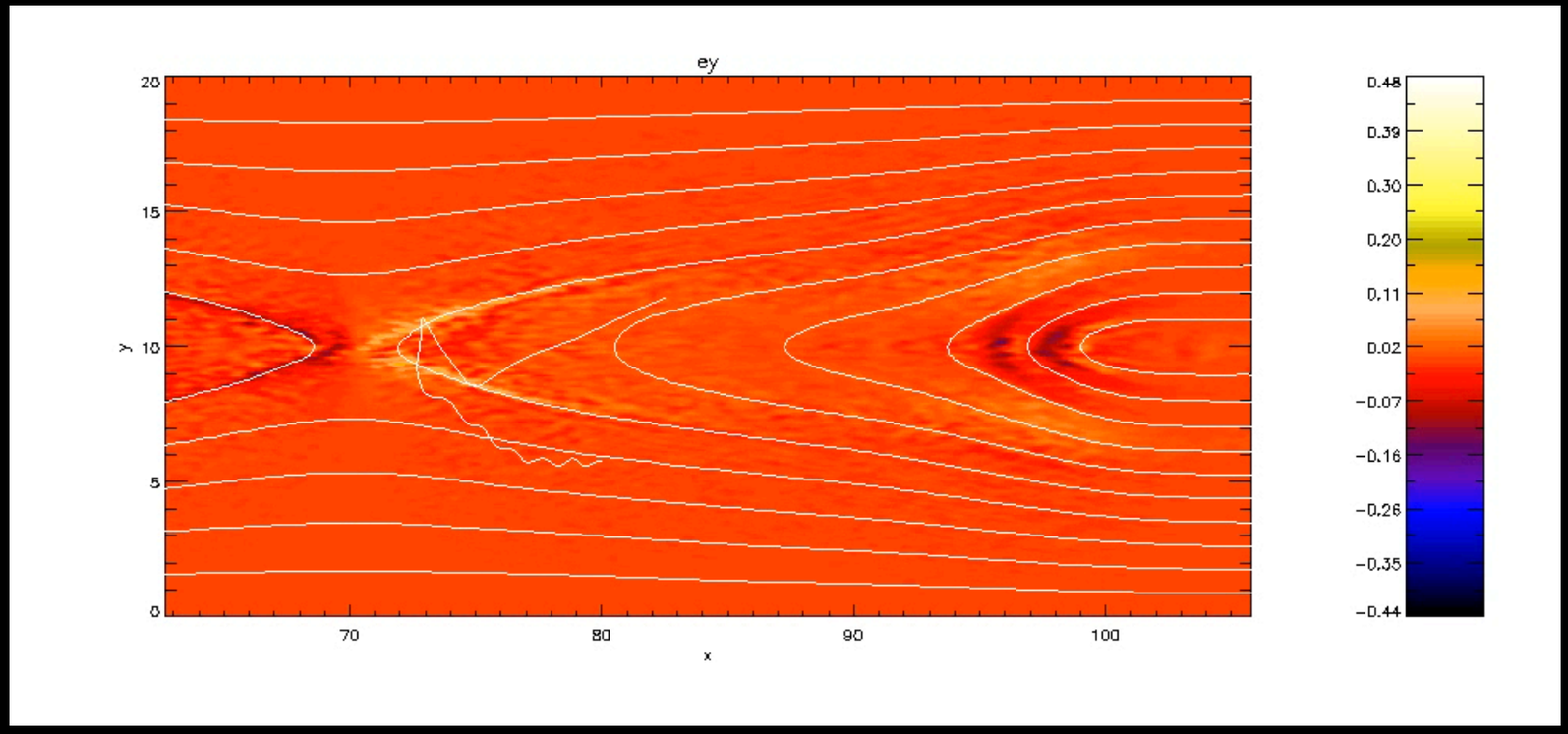
Variation Scales can be of the Order of Particle Larmor Scale



The bulk flow is « just » the local average of the particle velocities
Particles do **not** follow the same path

HALL MAGNETOHYDRODYNAMICS





The bulk flow is « just » the local average of the particle velocities
Particles do **not** follow the same path

VLASOV MODEL

Vlasov equation:

$$\frac{\partial f}{\partial t} + \mathbf{v} \cdot \frac{\partial f}{\partial \mathbf{r}} + \frac{q}{m} (\mathbf{E} + \mathbf{v} \times \mathbf{B}) \cdot \frac{\partial f}{\partial \mathbf{v}} = 0$$

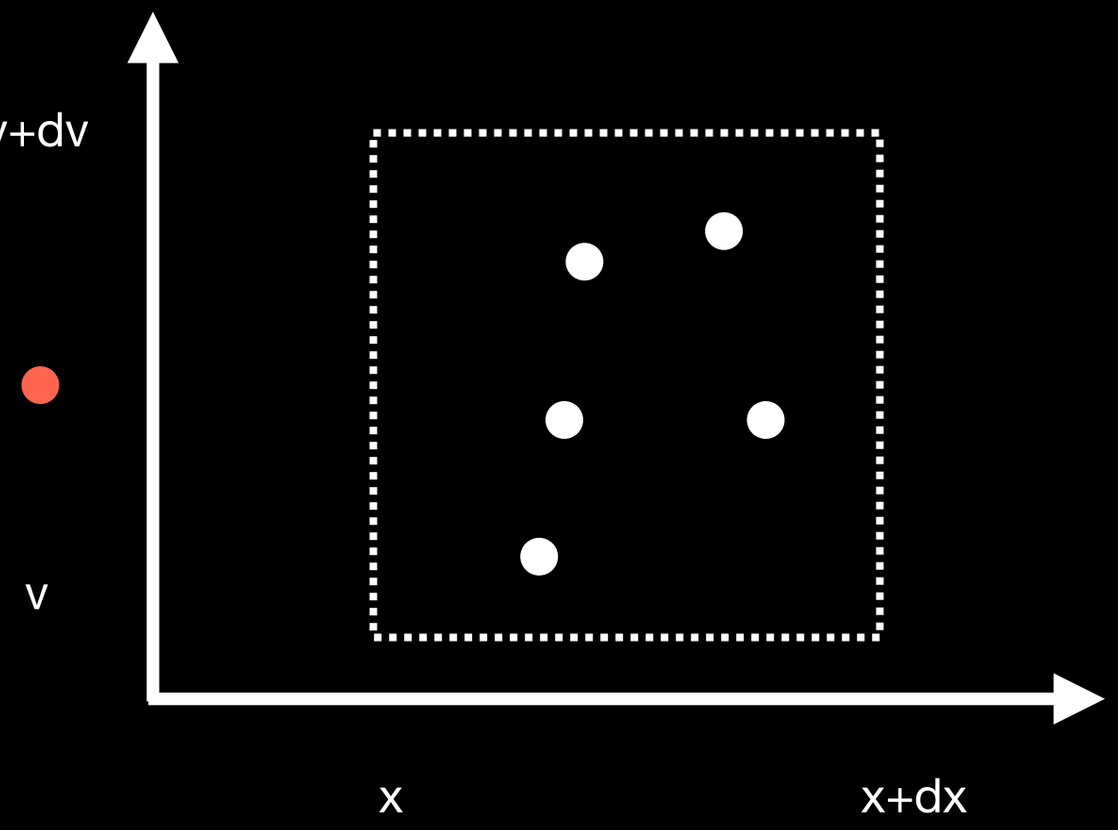
distribution functions of ions, electrons

$$f_i, f_e$$

Coupled to Maxwell Eq.

$$\frac{\partial E}{\partial t} = c^2 (\nabla \times \mathbf{B} - \mathbf{j})$$

$$\frac{\partial \mathbf{B}}{\partial t} = -\nabla \times \mathbf{E}$$



Vlasov equation:

$$\frac{\partial f}{\partial t} + \mathbf{v} \cdot \frac{\partial f}{\partial \mathbf{r}} + \frac{q}{m} (\mathbf{E} + \mathbf{v} \times \mathbf{B}) \cdot \frac{\partial f}{\partial \mathbf{v}} = 0$$

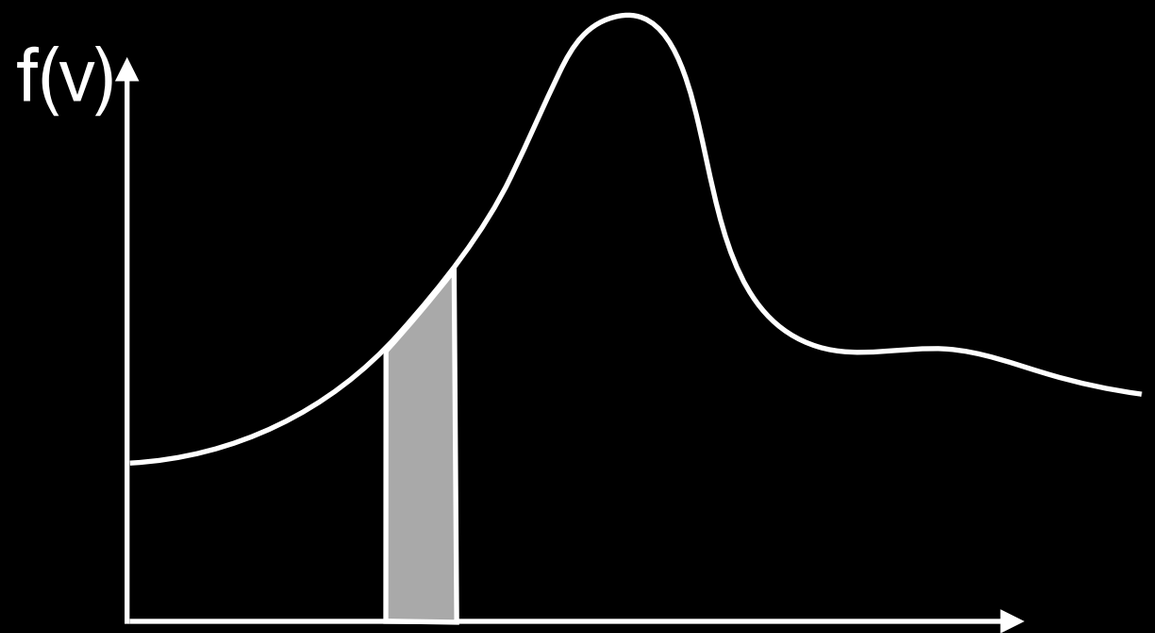
distribution functions of ions, electrons

$$f_i, f_e$$

Coupled to Maxwell Eq.

$$\frac{\partial E}{\partial t} = c^2 (\nabla \times \mathbf{B} - \mathbf{j})$$

$$\frac{\partial \mathbf{B}}{\partial t} = -\nabla \times \mathbf{E}$$



Number of particles at position r that have a velocity between v and v+dv

RECONNECTION AT THE MAGNETOPAUSE

Vlasov equation:

$$\frac{\partial f}{\partial t} + \mathbf{v} \cdot \frac{\partial f}{\partial \mathbf{r}} + \frac{q}{m} (\mathbf{E} + \mathbf{v} \times \mathbf{B}) \cdot \frac{\partial f}{\partial \mathbf{v}} = 0$$

distribution functions of ions, electrons

$$f_i, f_e$$

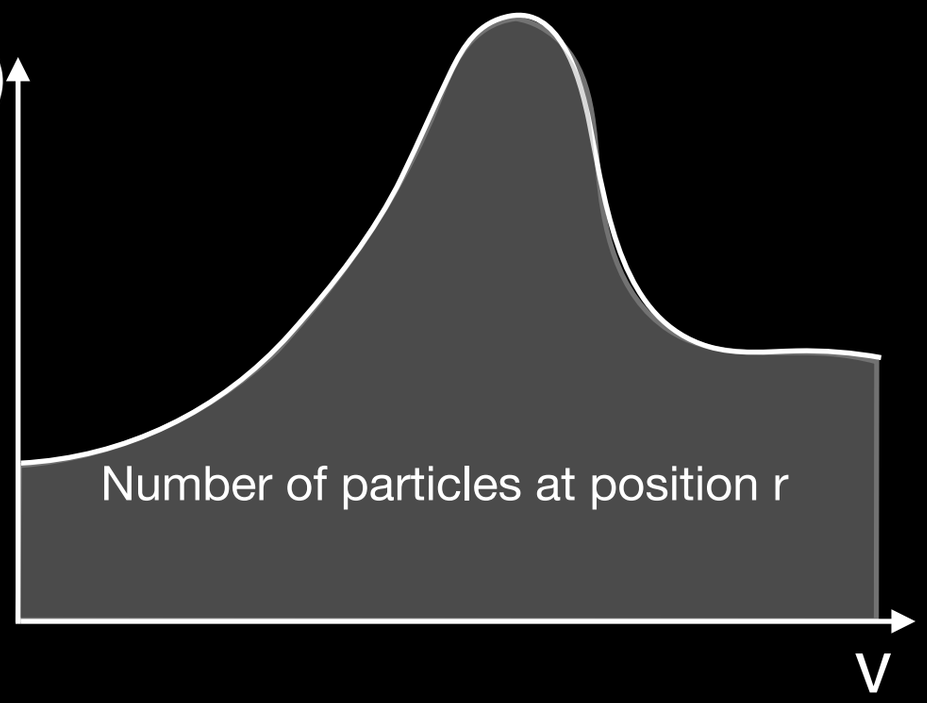
Coupled to Maxwell Eq.

$$\frac{\partial E}{\partial t} = c^2 (\nabla \times \mathbf{B} - \mathbf{j})$$

$$\frac{\partial \mathbf{B}}{\partial t} = -\nabla \times \mathbf{E}$$

« moments » of the distribution function

$$n(\mathbf{r}, t) = \int_{-\infty}^{\infty} f(\mathbf{r}, \mathbf{v}, t)\, d\mathbf{v}$$



RECONNECTION AT THE MAGNETOPAUSE

Vlasov equation:

$$\frac{\partial f}{\partial t} + \mathbf{v} \cdot \frac{\partial f}{\partial \mathbf{r}} + \frac{q}{m} (\mathbf{E} + \mathbf{v} \times \mathbf{B}) \cdot \frac{\partial f}{\partial \mathbf{v}} = 0$$

distribution functions of ions, electrons

$$f_i, f_e$$

Coupled to Maxwell Eq.

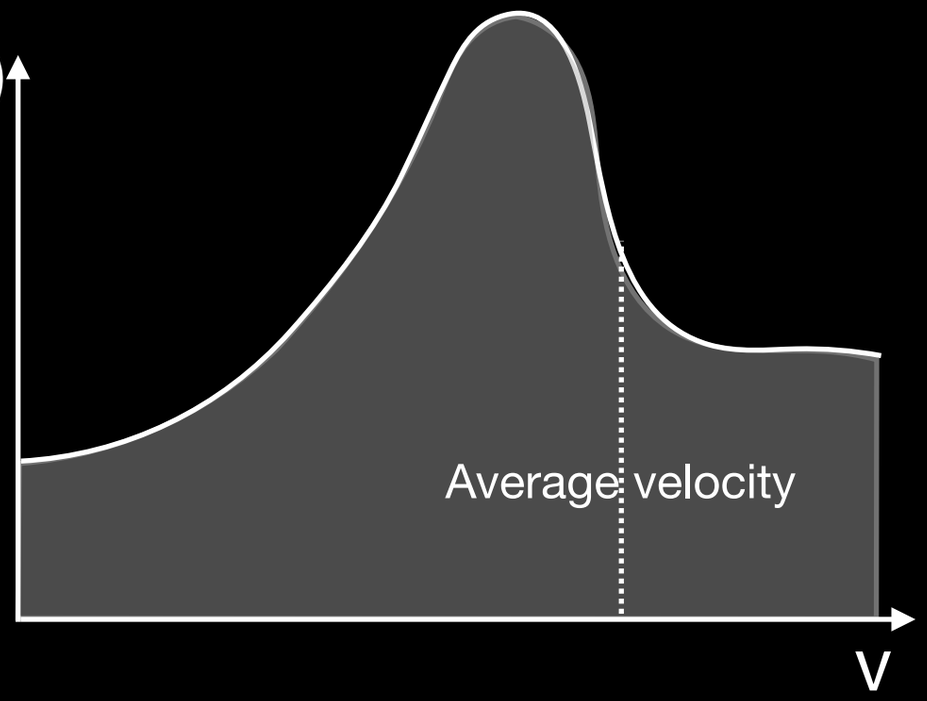
$$\frac{\partial E}{\partial t} = c^2 (\nabla \times \mathbf{B} - \mathbf{j})$$

$$\frac{\partial \mathbf{B}}{\partial t} = -\nabla \times \mathbf{E}$$

« moments » of the distribution function

$$n(\mathbf{r}, t) = \int_{-\infty}^{\infty} f(\mathbf{r}, \mathbf{v}, t)\, d\mathbf{v}$$

$$\mathbf{u}(\mathbf{r}, t) = \frac{1}{n} \int_{-\infty}^{\infty} \mathbf{v} f(\mathbf{r}, \mathbf{v}, t)\, d\mathbf{v}$$



RECONNECTION AT THE MAGNETOPAUSE

Vlasov equation:

$$\frac{\partial f}{\partial t} + \mathbf{v} \cdot \frac{\partial f}{\partial \mathbf{r}} + \frac{q}{m} (\mathbf{E} + \mathbf{v} \times \mathbf{B}) \cdot \frac{\partial f}{\partial \mathbf{v}} = 0$$

distribution functions of ions, electrons

$$f_i, f_e$$

Coupled to Maxwell Eq.

$$\frac{\partial E}{\partial t} = c^2 (\nabla \times \mathbf{B} - \mathbf{j})$$

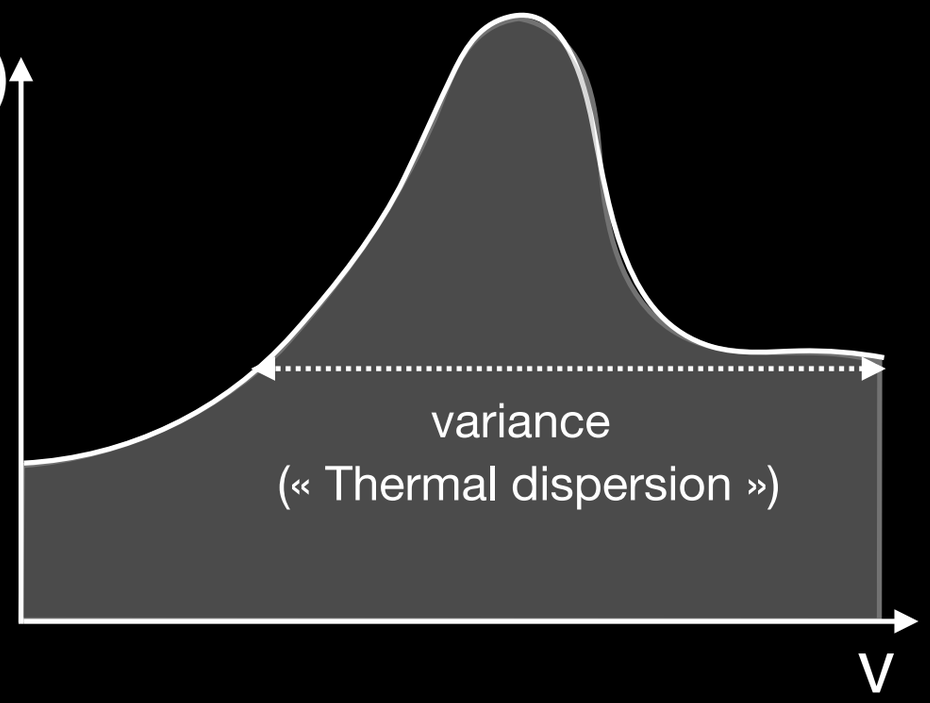
$$\frac{\partial \mathbf{B}}{\partial t} = -\nabla \times \mathbf{E}$$

« moments » of the distribution function

$$n(\mathbf{r}, t) = \int_{-\infty}^{\infty} f(\mathbf{r}, \mathbf{v}, t)\, d\mathbf{v}$$

$$\mathbf{u}(\mathbf{r}, t) = \frac{1}{n} \int_{-\infty}^{\infty} \mathbf{v} f(\mathbf{r}, \mathbf{v}, t)\, d\mathbf{v}$$

$$\mathbf{P}(\mathbf{r}, t) = \int_{-\infty}^{\infty} (\mathbf{v} - \mathbf{u}) \otimes (\mathbf{v} - \mathbf{u})\, f(\mathbf{r}, \mathbf{v}, t)\, d\mathbf{v}$$



RECONNECTION AT THE MAGNETOPAUSE

Vlasov equation:

$$\frac{\partial f}{\partial t} + \mathbf{v} \cdot \frac{\partial f}{\partial \mathbf{r}} + \frac{q}{m} (\mathbf{E} + \mathbf{v} \times \mathbf{B}) \cdot \frac{\partial f}{\partial \mathbf{v}} = 0$$

distribution functions of ions, electrons

$$f_i, f_e$$

Coupled to Maxwell Eq.

$$\frac{\partial E}{\partial t} = c^2 (\nabla \times \mathbf{B} - \mathbf{j})$$

$$\frac{\partial \mathbf{B}}{\partial t} = -\nabla \times \mathbf{E}$$

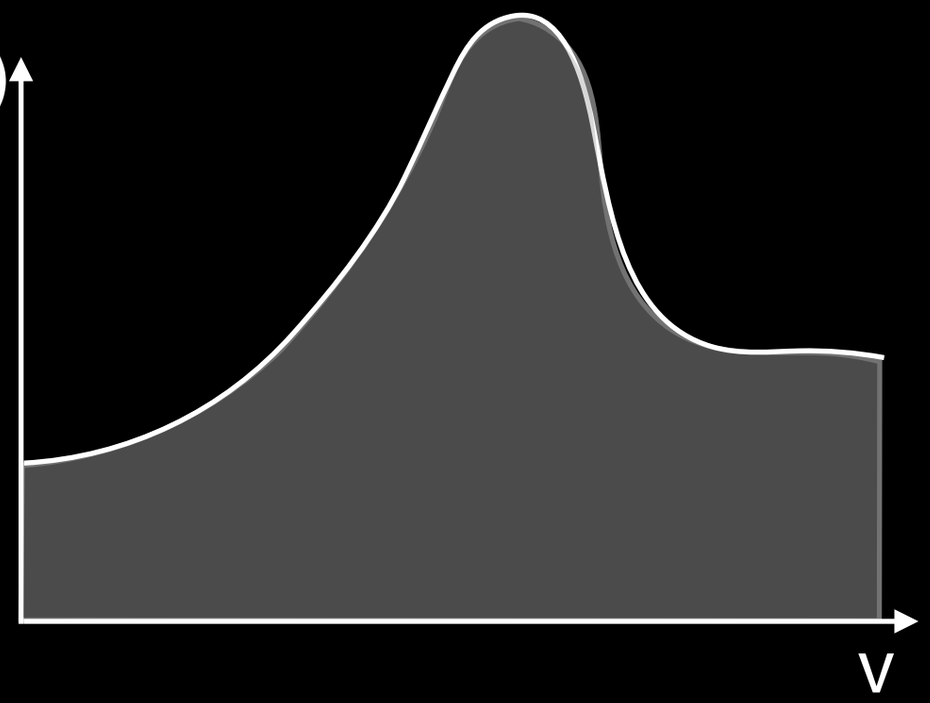
« moments » of the distribution function

$$n(\mathbf{r}, t) = \int_{-\infty}^{\infty} f(\mathbf{r}, \mathbf{v}, t)\, d\mathbf{v}$$

$$\mathbf{u}(\mathbf{r}, t) = \frac{1}{n} \int_{-\infty}^{\infty} \mathbf{v} f(\mathbf{r}, \mathbf{v}, t)\, d\mathbf{v}$$

$$\mathbf{P}(\mathbf{r}, t) = \int_{-\infty}^{\infty} (\mathbf{v} - \mathbf{u}) \otimes (\mathbf{v} - \mathbf{u})\, f(\mathbf{r}, \mathbf{v}, t)\, d\mathbf{v}$$

$$\mathbf{M}^n(\mathbf{r}, t) = \int_{-\infty}^{\infty} \mathbf{v}^n f(\mathbf{r}, \mathbf{v}, t)\, d\mathbf{v}$$



Endless number of moments…

RECONNECTION AT THE MAGNETOPAUSE

Vlasov equation:

$$\frac{\partial f}{\partial t} + \mathbf{v} \cdot \frac{\partial f}{\partial \mathbf{r}} + \frac{q}{m} (\mathbf{E} + \mathbf{v} \times \mathbf{B}) \cdot \frac{\partial f}{\partial \mathbf{v}} = 0$$

distribution functions of ions, electrons

$$f_i, f_e$$

Coupled to Maxwell Eq.

$$\frac{\partial E}{\partial t} = c^2 (\nabla \times \mathbf{B} - \mathbf{j})$$
$$\frac{\partial \mathbf{B}}{\partial t} = -\nabla \times \mathbf{E}$$

« moments » of the distribution function

$$n(\mathbf{r}, t) = \int_{-\infty}^{\infty} f(\mathbf{r}, \mathbf{v}, t)\, d\mathbf{v}$$

$$\mathbf{u}(\mathbf{r}, t) = \frac{1}{n} \int_{-\infty}^{\infty} \mathbf{v} f(\mathbf{r}, \mathbf{v}, t)\, d\mathbf{v}$$

$$\mathbf{P}(\mathbf{r}, t) = \int_{-\infty}^{\infty} (\mathbf{v} - \mathbf{u}) \otimes (\mathbf{v} - \mathbf{u})\, f(\mathbf{r}, \mathbf{v}, t)\, d\mathbf{v}$$

$$\mathbf{M}^n(\mathbf{r}, t) = \int_{-\infty}^{\infty} \mathbf{v}^n f(\mathbf{r}, \mathbf{v}, t)\, d\mathbf{v}$$

Endless number of moments…

Density depends on mass flux

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{v}) = 0$$

$$\rho \left(\frac{\partial \mathbf{v}}{\partial t} + \mathbf{v} \cdot \nabla \mathbf{v} \right) = -\nabla P + \mathbf{J} \times \mathbf{B}$$

momentum depends on pressure…

Fluid equations: truncated infinite system of moment equations.

VLASOV MODEL

$$\frac{\partial f}{\partial t} + \mathbf{v} \cdot \frac{\partial f}{\partial \mathbf{r}} + \frac{q}{m} (\mathbf{E} + \mathbf{v} \times \mathbf{B}) \cdot \frac{\partial f}{\partial \mathbf{v}} = 0$$

Evolve in time for all populations

$$n(\mathbf{r}, t) = \int_{-\infty}^{\infty} f(\mathbf{r}, \mathbf{v}, t)\, d\mathbf{v}$$

$$\mathbf{u}(\mathbf{r}, t) = \frac{1}{n} \int_{-\infty}^{\infty} \mathbf{v} f(\mathbf{r}, \mathbf{v}, t)\, d\mathbf{v}$$

Integrate in velocity space to get ion and electron densities and charge flux

$$\mathbf{j} = en(\mathbf{v} - \mathbf{v_e})$$

Calculate the electric current

$$\frac{\partial E}{\partial t} = c^2 (\nabla \times \mathbf{B} - \mathbf{j})$$

$$\frac{\partial \mathbf{B}}{\partial t} = -\nabla \times \mathbf{E}$$

Evolve Maxwell equations

Global Vlasov is insanely computationally heavy

$$\frac{\partial f}{\partial t} + \mathbf{v} \cdot \frac{\partial f}{\partial \mathbf{r}} + \frac{q}{m} (\mathbf{E} + \mathbf{v} \times \mathbf{B}) \cdot \frac{\partial f}{\partial \mathbf{v}} = 0$$

$$n(\mathbf{r}, t) = \int_{-\infty}^{\infty} f(\mathbf{r}, \mathbf{v}, t)\, d\mathbf{v}$$

$$\mathbf{u}(\mathbf{r}, t) = \frac{1}{n} \int_{-\infty}^{\infty} \mathbf{v} f(\mathbf{r}, \mathbf{v}, t)\, d\mathbf{v}$$

$$\mathbf{j} = en(\mathbf{v} - \mathbf{v_e})$$

$$\frac{\partial E}{\partial t} = c^2 (\nabla \times \mathbf{B} - \mathbf{j})$$

$$\frac{\partial \mathbf{B}}{\partial t} = -\nabla \times \mathbf{E}$$

$$f = f(\mathbf{r}, \mathbf{v}, t)$$

$$\underbrace{n_x n_y n_z}_{\text{Spatial domain}} \underbrace{n_{vx} n_{vy} n_{vz}}_{\text{Velocity domain}} \underbrace{n_t}_{\text{Temporal domain}}$$

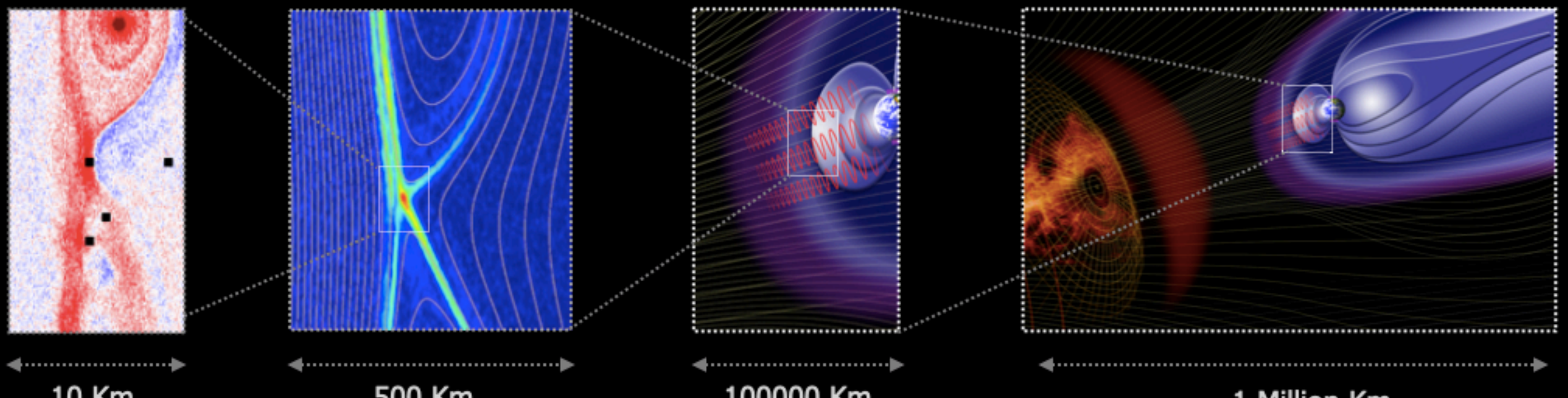
$$\Delta t \sim \Omega_{ce}^{-1} \approx 0.1\text{ms}, T \sim 1\text{hour} \rightarrow n_t \sim 4e7$$

$$\Delta x \sim \rho_e \approx 1\text{km}, L \sim 50\text{Re} \rightarrow n_x \sim 2e5$$

$$\Delta v \sim 0.01 V_A \approx 5\text{km/s}, V \sim 5 V_A \rightarrow n_v \sim 5e2$$

In a thousand years maybe…

What do simulations say? Pick a model…



Fully Kinetic

fully kinetic physics, but small domains, short durations

Fluid « MHD »

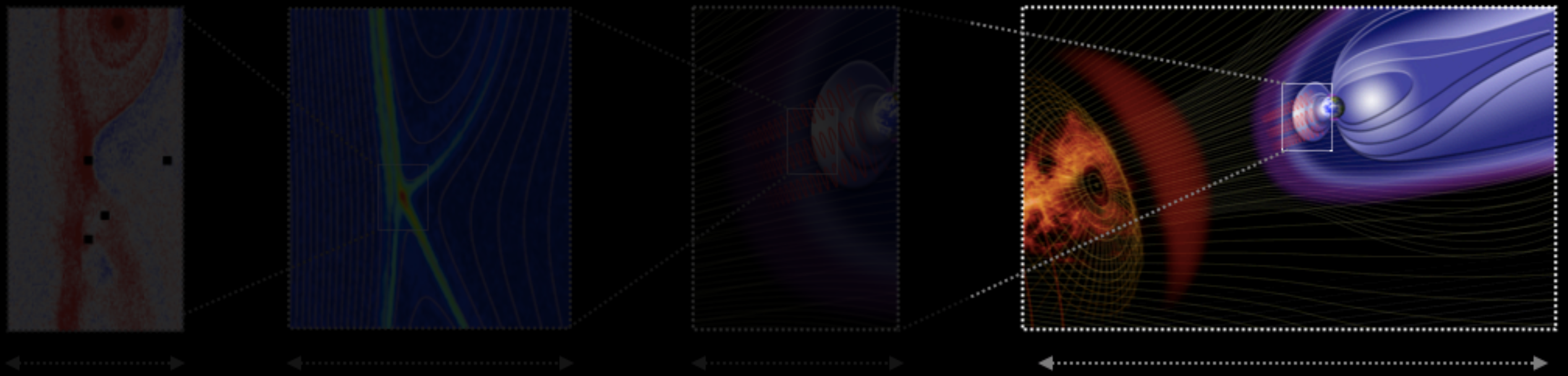
Global scale system
But no kinetic physics

ω_{ce}^{-1} ρ_e

ω_{ci}^{-1} ρ_i

L, T

What do simulations say? Pick a model…



Fully Kinetic

fully kinetic physics, but small domains, short durations

ω_{ce}^{-1} ρ_e

ω_{ci}^{-1} ρ_i

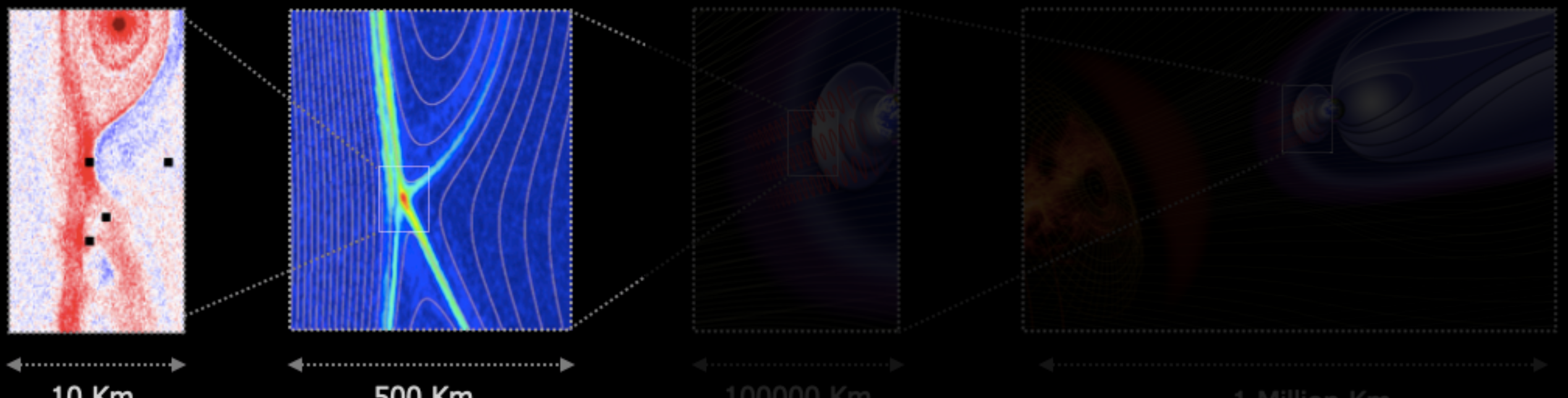
Fluid « MHD »

Global scale system

But no kinetic physics

L,T

What do simulations say? Pick a model…



Fully Kinetic

fully kinetic physics, but small domains, short durations

Fluid « MHD »

Global scale system

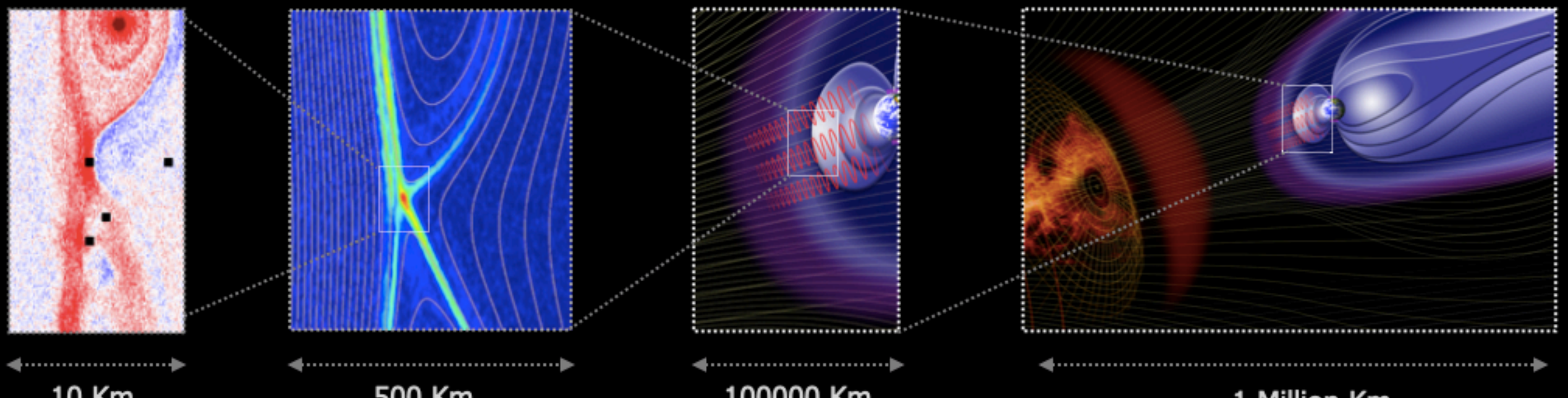
But no kinetic physics

ω_{ce}^{-1} ρ_e

ω_{ci}^{-1} ρ_i

L, T

What do simulations say? Pick a model…



Fully Kinetic

fully kinetic physics, but small domains, short durations

Fluid « MHD »

Global scale system
But no kinetic physics

ω_{ce}^{-1} ρ_e

ω_{ci}^{-1} ρ_i

L, T

Hybrid kinetic

include ion kinetics but a fluid electron model

Hybrid kinetic formalism

Faraday's law

$$\frac{\partial \mathbf{B}}{\partial t} = -\nabla \times \mathbf{E},$$

Ampere's law

$$\mu_0 \, \mathbf{j} = \nabla \times \mathbf{B},$$

Ion Vlasov eq.

$$\frac{\partial f_p}{\partial t} = -\mathbf{v} \cdot \nabla f_p - \frac{\mathbf{E} + \mathbf{v} \times \mathbf{B}}{m_p} \cdot \nabla_{\mathbf{v}} f_p,$$

Ion moments

$$n_i = \sum_p \int f_p(\mathbf{r}, \mathbf{v}) \, d^3 v,$$

$$\mathbf{v}_i = \frac{1}{n_i} \sum_p \int \mathbf{v} \, f_p(\mathbf{r}, \mathbf{v}) \, d^3 v,$$

Quasi-neutrality

$$n_i = n_e = n,$$

$$\mathbf{v}_e = \mathbf{v}_i - \frac{\mathbf{j}}{ne},$$

Electron momentum eq.

$$m_e n_e \frac{d\mathbf{v_e}}{dt} = -\nabla \cdot P_e - e n_e \left(\mathbf{E} + \mathbf{v_e} \times \mathbf{B} \right)$$

Hybrid kinetic formalism

Faraday's law

$$\frac{\partial \mathbf{B}}{\partial t} = -\nabla \times \mathbf{E},$$

Ampere's law

$$\mu_0 \, \mathbf{j} = \nabla \times \mathbf{B},$$

Ion Vlasov eq.

$$\frac{\partial f_p}{\partial t} = -\mathbf{v} \cdot \nabla f_p - \frac{\mathbf{E} + \mathbf{v} \times \mathbf{B}}{m_p} \cdot \nabla_{\mathbf{v}} f_p,$$

Ion moments

$$n_i = \sum_p \int f_p(\mathbf{r}, \mathbf{v}) \, d^3 v,$$

$$\mathbf{v}_i = \frac{1}{n_i} \sum_p \int \mathbf{v} \, f_p(\mathbf{r}, \mathbf{v}) \, d^3 v,$$

Quasi-neutrality

$$n_i = n_e = n,$$

$$\mathbf{v}_e = \mathbf{v}_i - \frac{\mathbf{j}}{ne},$$

Generalized Ohm's law

$$\mathbf{E} = -\mathbf{v}_e \times \mathbf{B} - \frac{1}{en} \nabla \cdot \mathbf{P}_e - \frac{m_e}{e} \frac{d\mathbf{v}_e}{dt}.$$

Kinetic for ions, fluid for electrons.

This is what you will implement :-)

3 STEPS ITERATED CRANK NICHOLSON

And you will implement it this way ;-)

t

Prediction

$$\mathbf{B}_{p1}^{n+1} = \mathbf{B}^n - \Delta t \nabla \times \mathbf{E}^n$$

$$\mathbf{E}_{p1}^{n+1} = -\mathbf{u}^n \times \mathbf{B}_{p1}^{n+1} + \frac{\nabla \times \mathbf{B}_{p1}^{n+1}}{N^n} - \frac{\nabla \cdot \mathbf{P_e}}{N^n} + \eta \nabla \times \mathbf{B}_{p1}^{n+1} - \nu \nabla^2 \nabla \times \mathbf{B_{p1}^{n+1}}$$

$$(\mathbf{E}, \mathbf{B})^{n+1/2} =< (\mathbf{E}, \mathbf{B}) >_n^{n+1}$$

$$\mathbf{r}_{p1}^{n+1/2} = \mathbf{r}^n + \Delta t/2 \mathbf{v}^n$$

$$\mathbf{E}, \mathbf{B}\left(\mathbf{r_{p1}^{n+1/2}}\right) = \sum_{ijk} \left(\mathbf{E}, \mathbf{B}_{ijk}\right) W\left(|\mathbf{r}_{ijk} - \mathbf{r}_{p1}^{n+1/2}|\right)$$

$$m_i \frac{d\mathbf{v}_{p1}^{n+1}}{dt} = e\left(\mathbf{v}^n \times + \mathbf{B}^{n+1/2} + \mathbf{E}^{n+1/2}\right)$$

$$N^{n+1} = \sum_p w_p \mathbf{v}_{p1}^{n+1} W\left(|\mathbf{r}_{ijk} - \mathbf{r}_{p1}^{n+1}|\right) \qquad u^{n+1} = \sum_p w_p \mathbf{v}_{p1}^{n+1} W\left(|\mathbf{r}_{ijk} - \mathbf{r}_{p1}^{n+1}|\right)$$

Prediction

$$\mathbf{B}_{p2}^{n+1} = \mathbf{B}^n - \Delta t \nabla \times \mathbf{E}^{n+1/2}$$

$$\mathbf{E}_{p2}^{n+1} = -\mathbf{u}^{n+1} \times \mathbf{B}_{p2}^{n+1} + \frac{\nabla \times \mathbf{B}_{p2}^{n+1}}{N^{n+1}} - \frac{\nabla \cdot \mathbf{P_e}}{N^{n+1}} + \eta \nabla \times \mathbf{B}_{p2}^{n+1} - \nu \nabla^2 \nabla \times \mathbf{B_{p2}^{n+1}}$$

$$\mathbf{r}_{p2}^{n+1/2} = \mathbf{r}^n + \Delta t/2 \mathbf{v}^n$$

$$(\mathbf{E}, \mathbf{B})^{n+1/2} =< (\mathbf{E}, \mathbf{B}) >_n^{n+1}$$

$$m_i \frac{d\mathbf{v}_{p2}^{n+1}}{dt} = e\left(\mathbf{v}^n \times + \mathbf{B}^{n+1/2} + \mathbf{E}^{n+1/2}\right)$$

$$N^{n+1} = \sum_p w_p \mathbf{v}_{p1}^{n+1} W\left(|\mathbf{r}_{ijk} - \mathbf{r}_{p1}^{n+1}|\right) \qquad u^{n+1} = \sum_p w_p \mathbf{v}_{p1}^{n+1} W\left(|\mathbf{r}_{ijk} - \mathbf{r}_{p1}^{n+1}|\right)$$

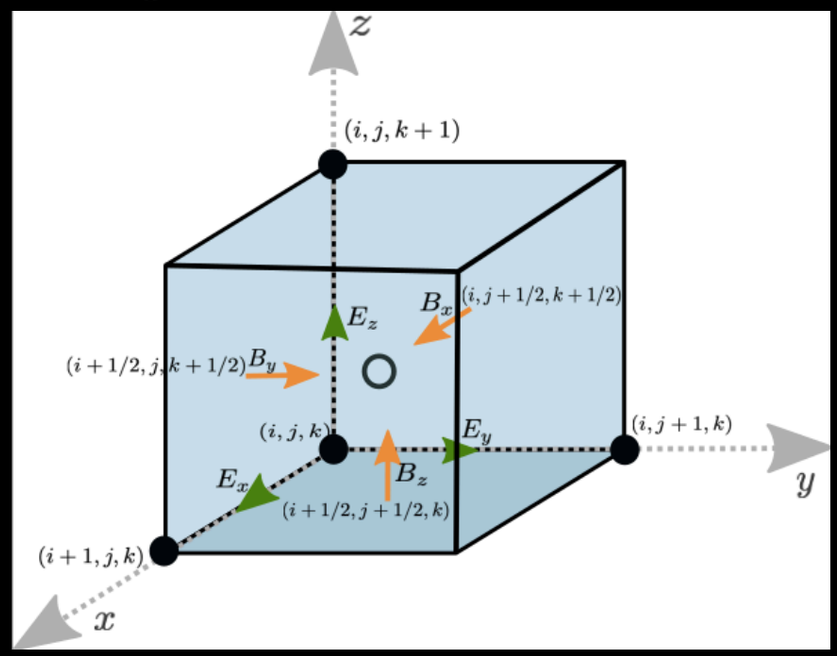
Correction

$$\mathbf{B}^{n+1} = \mathbf{B}^n - \Delta t \nabla \times \mathbf{E}^{n+1/2}$$

$$\mathbf{E}^{n+1} = -\mathbf{u}^{n+1} \times \mathbf{B}^{n+1} + \frac{\nabla \times \mathbf{B}^{n+1}}{N^{n+1}} - \frac{\nabla \cdot \mathbf{P_e}}{N^{n+1}} + \eta \nabla \times \mathbf{B}^{n+1} - \nu \nabla^2 \nabla \times \mathbf{B^{n+1}}$$

$t + \Delta t$

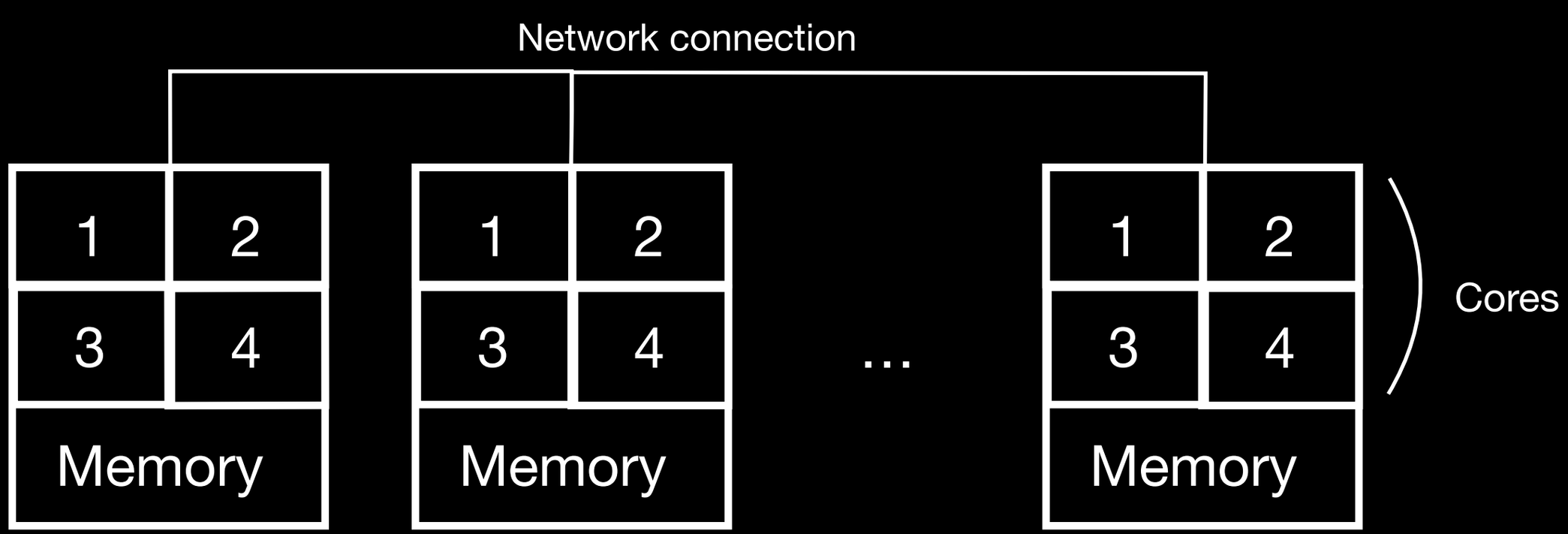
Yee grid



Some elements for high performance



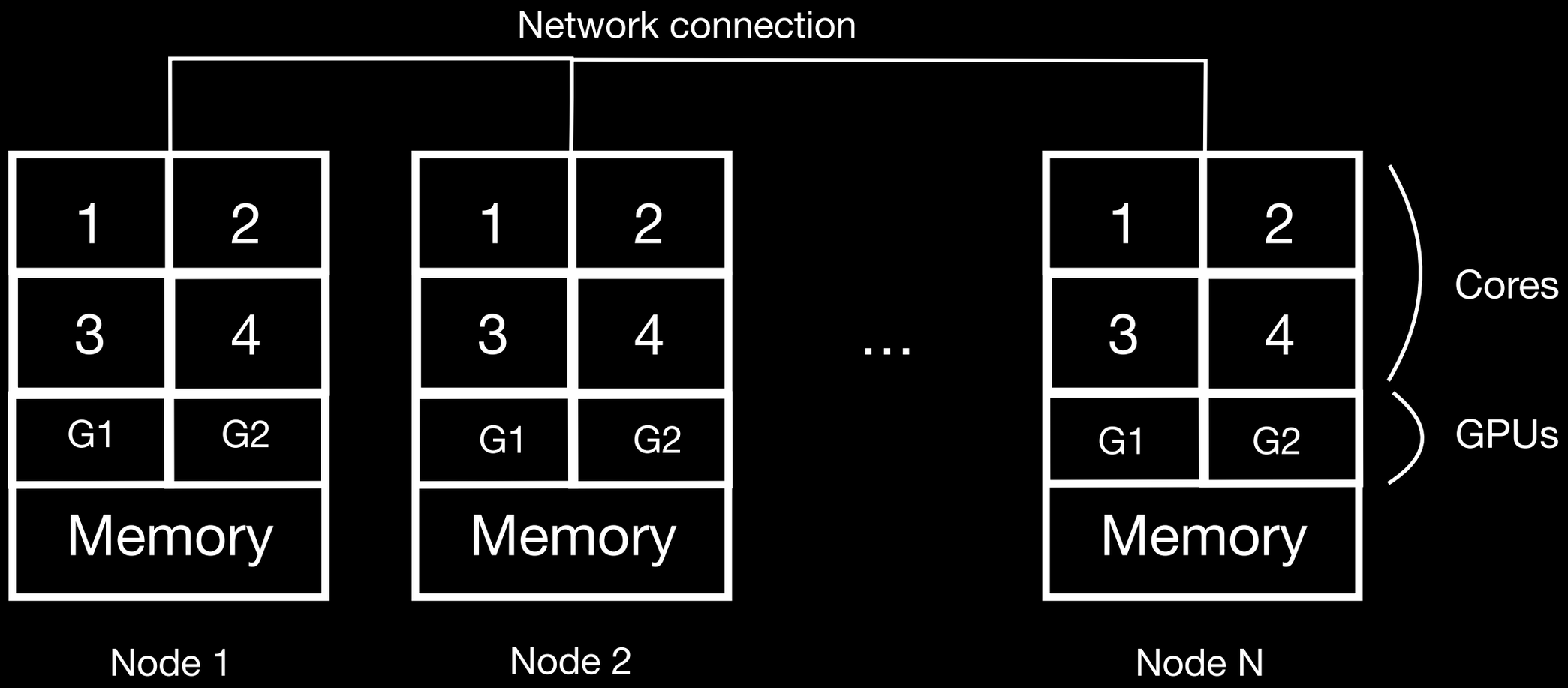
SUPERCOMPUTER ARCHITECTURE



« Nodes » are linked via fast network connections
Each node has its own memory

Each node has several « cores » sharing the same memory

Modern (heterogeneous) Supercomputer architecture



« Nodes » are linked via fast network connections
Each node has its own memory

Each node has several « cores » sharing the same memory

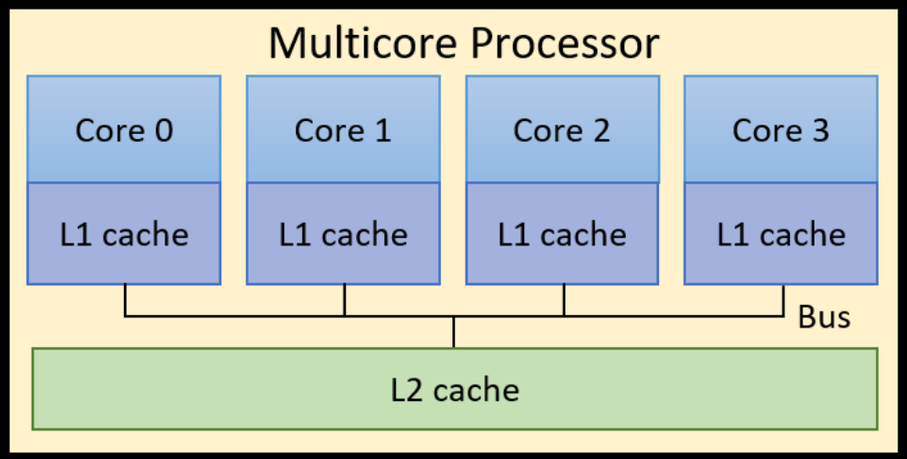
Each node may have one or several GPUs

SUPERCOMPUTER @ GENCI

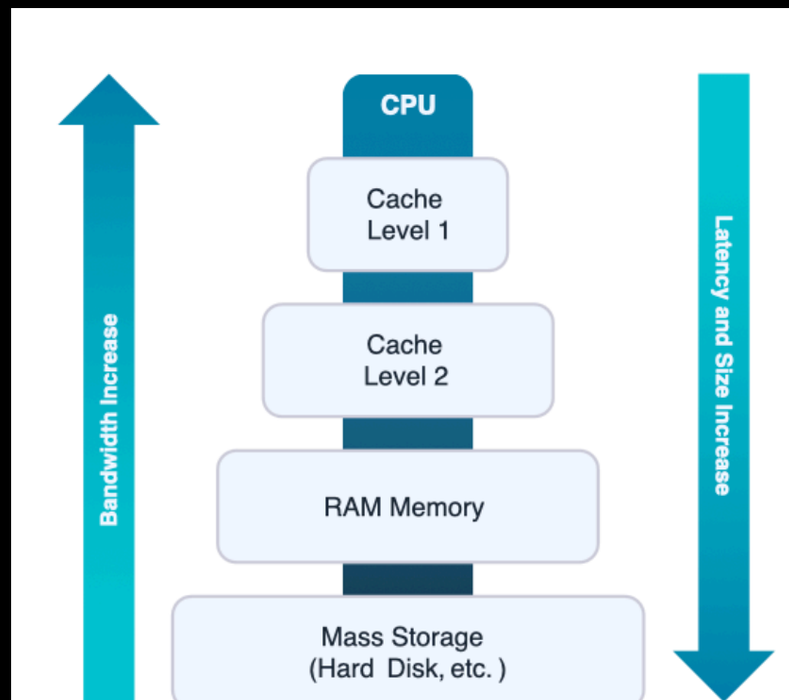
GENCI : Grand Equipement National Calcul Intensif

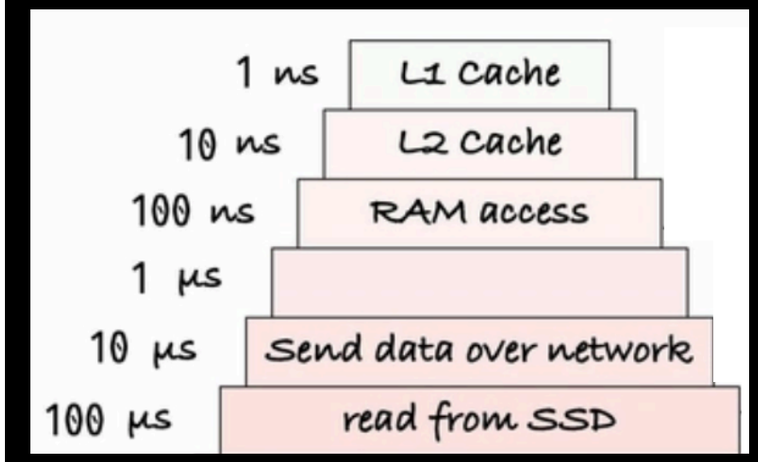
- https://www.genci.fr/centre-informatique-national-de-lenseignement-superieur-cines
- https://www.genci.fr/institut-du-developpement-et-des-ressources-en-informatique-scientifique-idris
- https://www.genci.fr/tres-grand-centre-de-calcul-du-cea-tgcc

MULTICORE PROCESSOR MEMORY

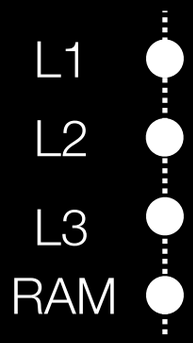


Main Memory (RAM)





- Loading data and instruction from memory is expensive
- Memory is decomposed into a « cache hierarchy »

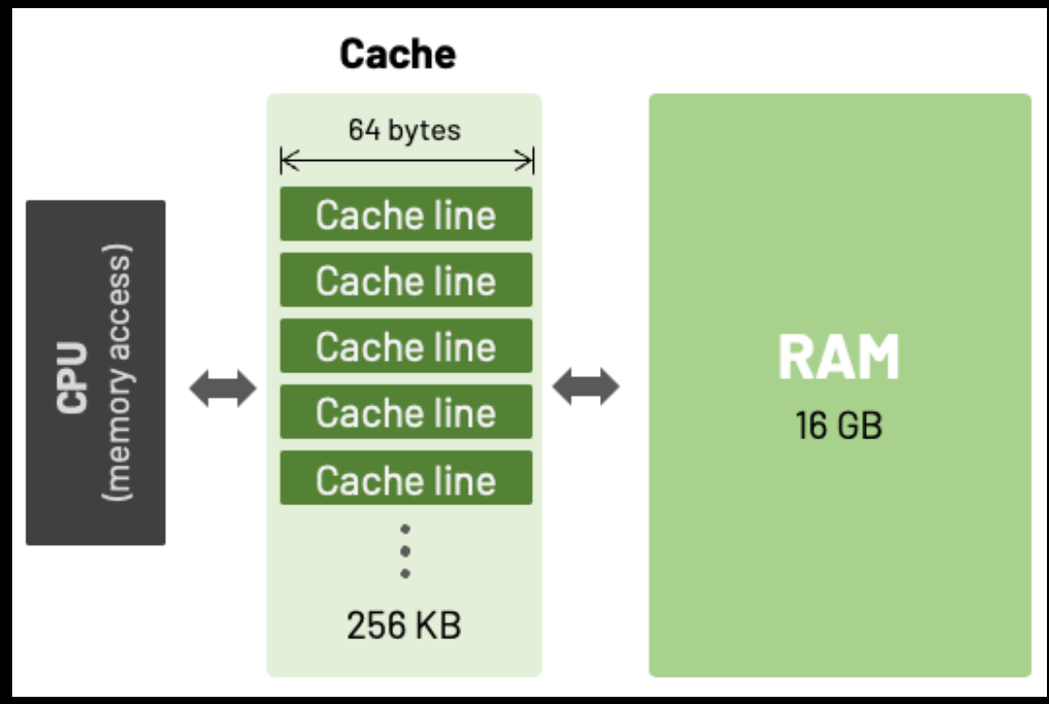


In memory

CPU

Multicore processor memory

- Cache is not a contiguous block of memory
- It is made of a set of *cache lines* of a fixed size
- Data is fetched for an entire cache line
- Contiguous memory populate cache line
- « oldest » (usage) cache lines are replaced first



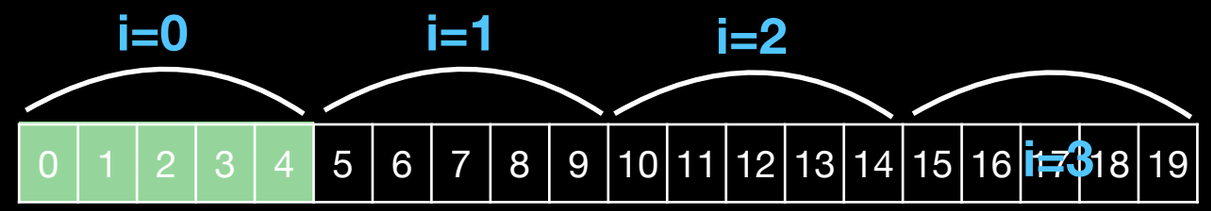
Both data and instructions are pre-fetched into the cache to minimize the probability of cache misses

Multicore processor memory

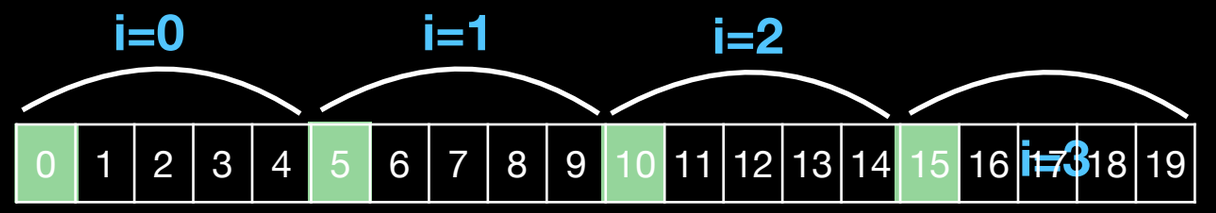
```
for (int i=; i < 4; ++i)
{
    for (int j=0; j < 5; ++j)
    {
        field[i][j] = ...;
    }
}
```

```
for (int j=; j < 5; ++j)
{
    for (int i=0; i < 4; ++i)
    {
        field[i][j] = ...;
    }
}
```

	i=0	i=1	i=2	i=3
j=0	0	5	10	15
j=1	1	6	11	16
j=2	2	7	12	17
j=3	3	8	13	18
j=4	4	9	14	19



Next index probably in cache



Next index probably not in cache…

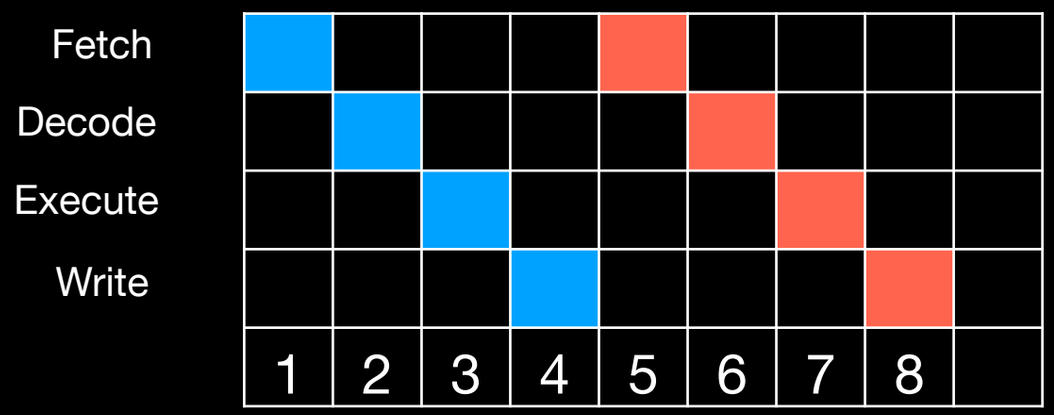
- « Cache hit » : required data is in cache, low latency
- « Cache miss »: data not in cache… you wait

- Be careful of how you traverse your data!
- Worst is probably random accesses….
- Beware spatial and temporal coherence

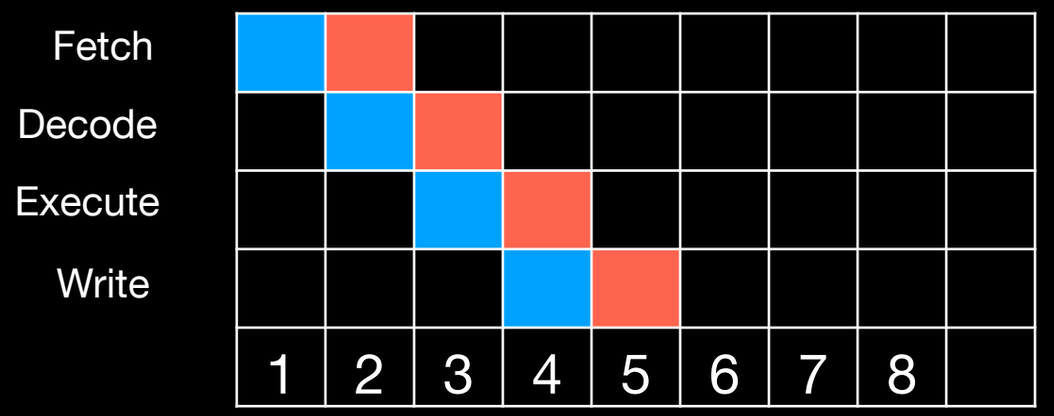
Memory bound programs: run time dominated by memory accesses. Huge arrays and little operations per element
CPU bound programs: huge numbers of heavy computations for little number of elements

CPU PIPELINE

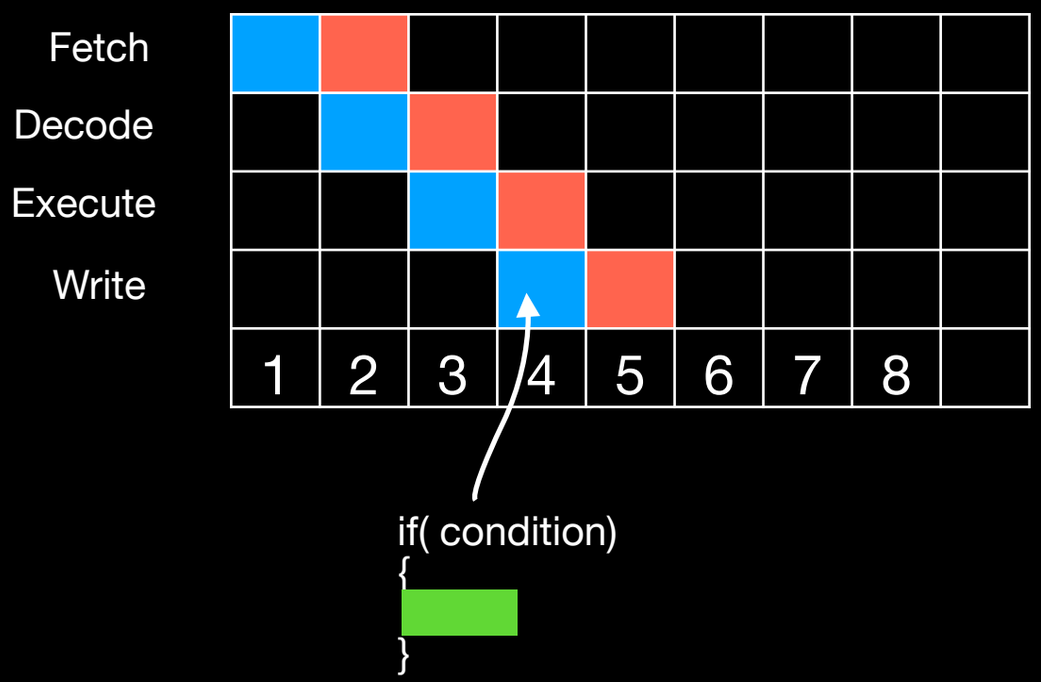
Two instructions without pipelining

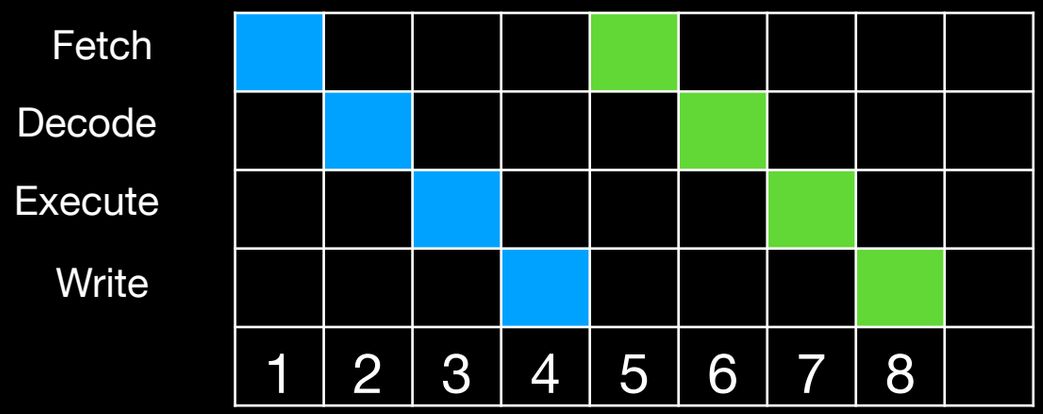


Two instructions with pipelining



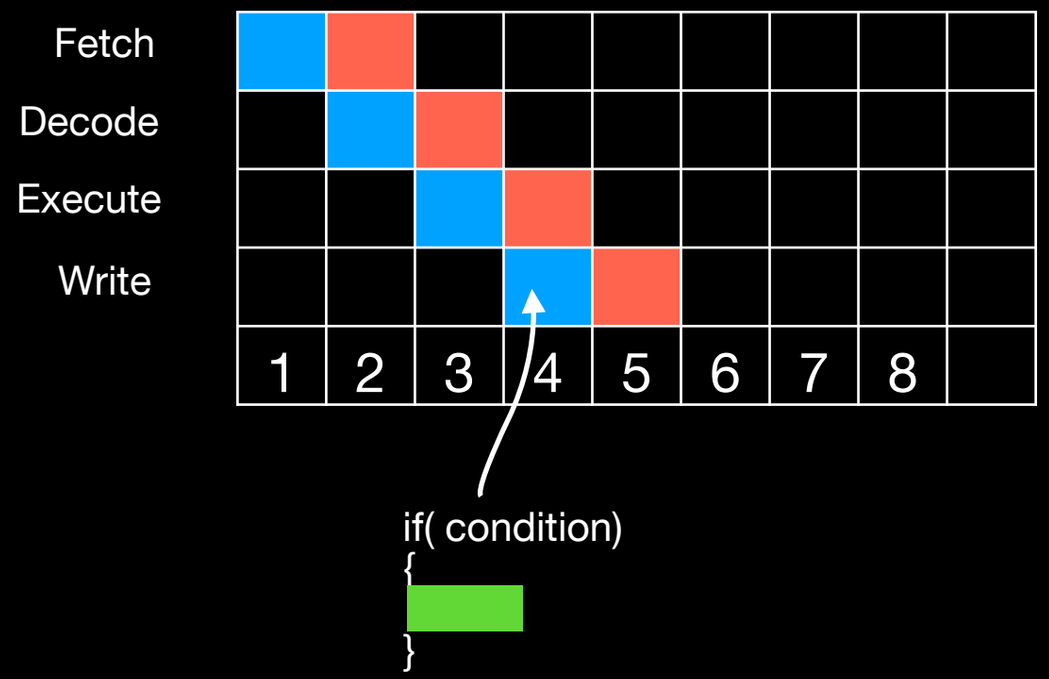
Handling an instruction requires several steps that the CPU manages in a « pipeline »

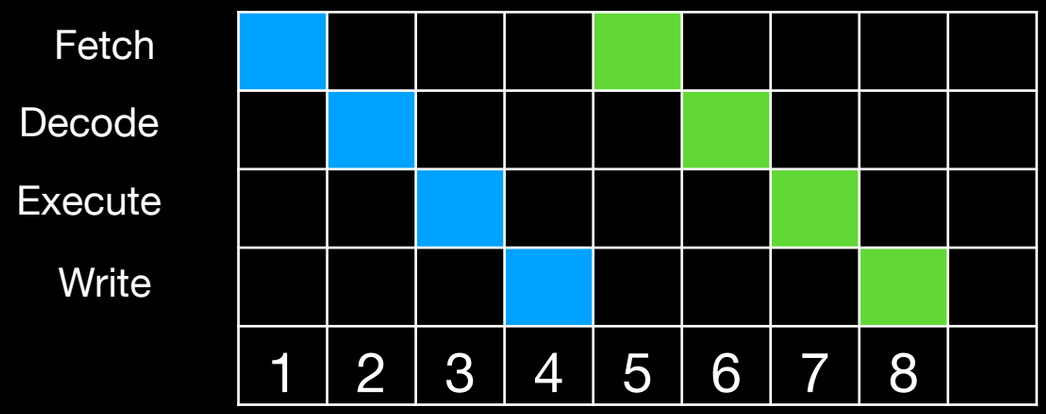




« branching » potentially kills the flow in the pipeline and wastes cycles.

CPU PIPELINE





« branching » potentially kills the flow in the pipeline and wastes cycles.

Avoid conditions and virtual functions in « heavy » loops

- **Out-of-order** pipelines: CPUs are « smart » enough to handle *independent* instructions to fill the pipeline
- **Superscalar** CPUs: instructions are dispatched to several execution units working in parallel
- **Branch prediction**: the CPU tries its best to identify *patterns* and predict the next instructions

Compilers are very smart. Gcc is ~15 million lines of codes…

General optimization strategy

Premature optimization is the root of all evil… but don't be dumb!

- Make it work
- Make it clean
- Make it fast

Measure measure measure…..

Very hard to predict what's taking time…

- Implement your own timer
- Use tools
  - perf: https://perfwiki.github.io/main/
  - gprof: https://hpc-wiki.info/hpc/Gprof_Tutorial
  - vtune (intel)
  - etc.

https://coliru.stacked-crooked.com/a/357dc3307f794823

```C++
template<typename Fn>
auto measure(Fn fn)
{

    std::vector<double> durations;
    int const repeatTimes = 100;
    auto ret = 0;

    for (int step = 0; step < repeatTimes; ++step)
    {

        std::chrono::high_resolution_clock::time_point t1;
        t1 = std::chrono::high_resolution_clock::now();


        ret = fn();

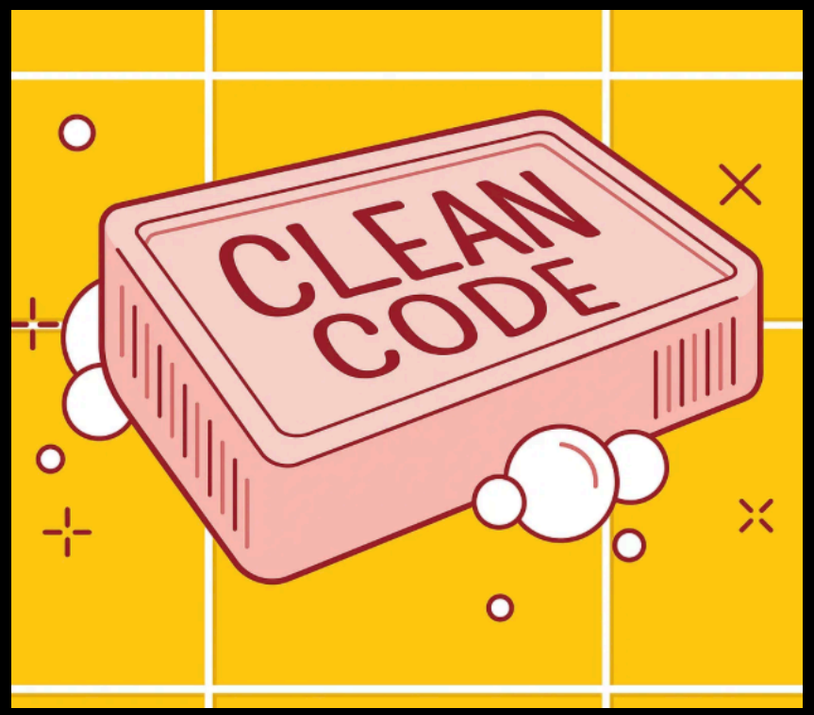

        std::chrono::high_resolution_clock::time_point t2;
        t2 = std::chrono::high_resolution_clock::now();

        auto duration = std::chrono::duration_cast<std::chrono::nanoseconds>(t2 - t1
).count();
        durations.push_back(duration);


    }
    std::cout << "use the value : " << ret  << "\n";

    return std::accumulate(std::begin(durations), std::end(durations),
0.0)/repeatTimes;
}
```

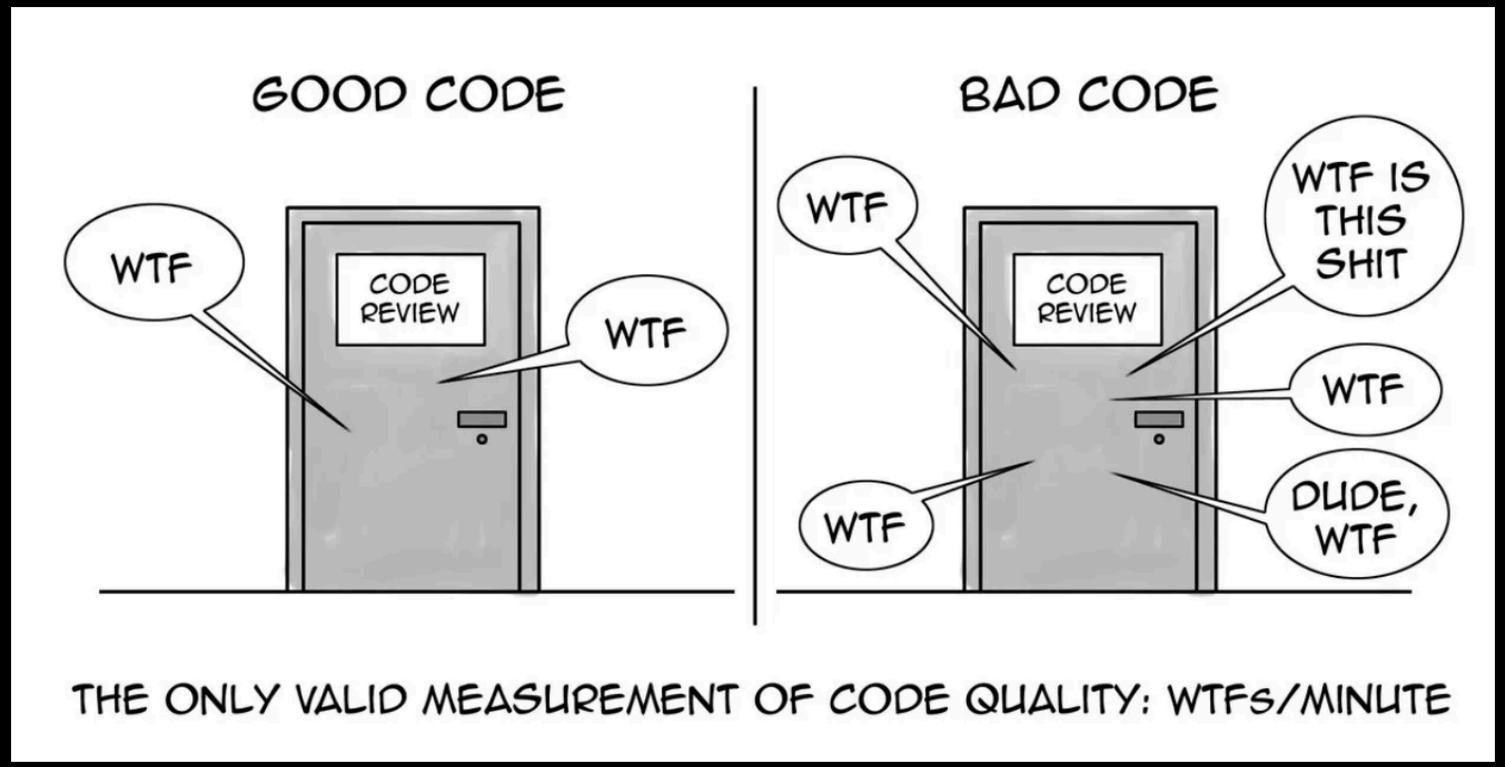
1. Choose the « best » algorithm
2. Make it faster
3. Parallelization


CLEAN
CODE

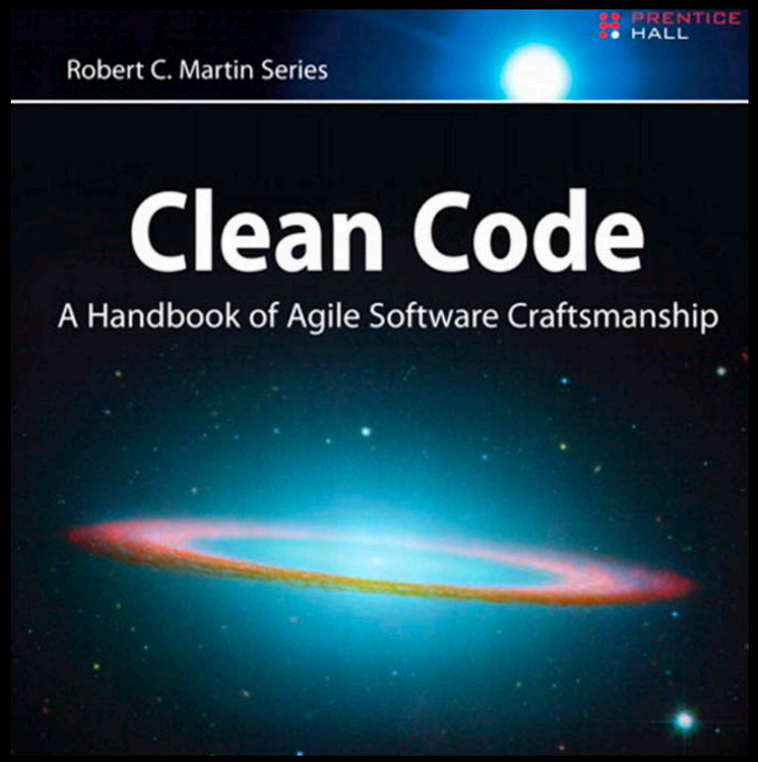
CODE IS FOR *HUMANS*

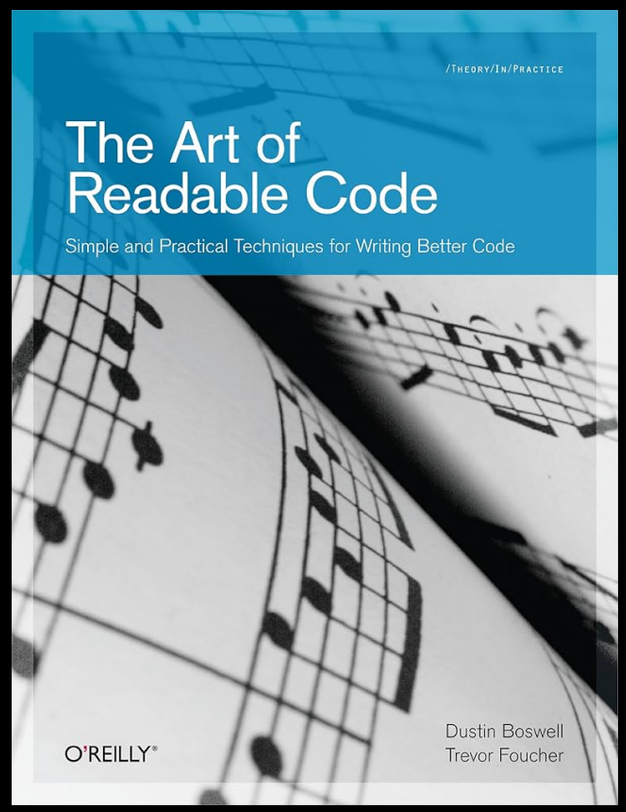
All of your projects will always have at least 2 developers

You, and you 6 months later… help yourself!



SOME REFERENCES





Naming things correctly

- Naming things is hard, it takes time, it’s normal, rush now: lose time later

- Choose descriptive and unambiguous names
- Follow standard conventions
- Use pronounceable names (read your code like English)
- Use searchable names
- Replace magic numbers with named constants
- Don't append prefixes or type information

Naming things correctly

- Naming things is hard, it takes time, it's normal, rush now: lose time later
- Functions should have no arguments, or one at most. Two is one too many probably. 3 is code smell

Say you want a piece of code that adjusts the time step if the CFL condition is violated

```
if (0.9*v/(dx/dt) < 1)
{
    // adjust time step
}
```

- What the f**** does it mean?

```
if (!stability_criterion(v, dt, dx))
{
    // adjust time step
}
```

- Better but does not read well…
- « not stability criterion »? What does it mean?
- dx : limited to 1D?
- Should we know at this level that we need v, dt and dx?

```
if (!isStable(v, dt, dx))
{
    // adjust time step
}
```

- Reads better
- Same comments as above for the rest

```
if (!simulation.isStable(solution_state))
{
    // adjust time step
}
```

- Best?

Respect the level of abstraction

You start your car to go somewhere:

- Open the door
  - Door unlocking mechanism…
- Sit down
- Push start (turn key?)
- Release brakes
  - brakes move away from disks…
- Push gas pedal
  - Gas flow into engine
  - Gas mixes with air in combustion chamber
- Drive

- Open the door
- Sit down
- Push start (turn key?)
- Release break
- Push gas pedal and drive

Only write in the current function <u>what is relevant to the current context</u>

Respect the level of abstraction

You have a grid and want, somewhere, to get the coordinates of each nodes

What do you do?

Typically, you write this function:

```
auto get_coordinates_from_grid(std::vector<double> const& grid, int i, int j)
{
    ///
}
```

What's wrong with it?

Respect the level of abstraction

You have a grid and want, somewhere, to get the coordinates of each nodes

```
auto get_coordinates_from_grid(std::vector<double> const& grid, int i, int j)
{
    ///
}
```

Call site:

```
for (int i = 0; i < grid_size; ++i)
{
    for (int j = 0; j < grid_size; ++j)
    {
        auto coords = get_coordinates_from_grid(my_grid, i, j);
    }
}
```

- The function leaks that the grid is a vector of double, and it is 2d
- Call site needs to handle the 2Dness, the grid size, and (i,j)
- « grid » is needlessly repeated
- « get » is useless, we see we get something from coords =

Do NOT write low level first. **Start high level and go down.**

Respect the level of abstraction

Call site:

```
    std::size_t nx = 10, ny = 20;;
    Grid my_grid{nx, ny};;
    for (auto const& idx: my_grid)
    {
        auto coords = my_grid.coords_at(idx);
    }

}
```

- ‘coords_at’: enough, ‘idx’ complements
- Who cares how Grid works internally here?
- Easily iterate over a grid with range-based loop

Writing the high level first:

- fixes the interface of objects
- helps respecting the right level of abstraction
- Simpler code that reads more easily, less bugs, better extensibility, etc.
- More difficult to get overwhelmed by details…

Respect the level of abstraction

t

Prediction

$$\mathbf{B}_{p1}^{n+1} = \mathbf{B}^n - \Delta t \nabla \times \mathbf{E}^n$$

$$\mathbf{E}_{p1}^{n+1} = -\mathbf{u}^n \times \mathbf{B}_{p1}^{n+1} + \frac{\nabla \times \mathbf{B}_{p1}^{n+1}}{N^n} - \frac{\nabla \cdot \mathbf{P_e}}{N^n} + \eta \nabla \times \mathbf{B}_{p1}^{n+1} - \nu \nabla^2 \nabla \times \mathbf{B}_{\mathbf{p1}}^{\mathbf{n+1}}$$

$$(\mathbf{E}, \mathbf{B})^{n+1/2} =< (\mathbf{E}, \mathbf{B}) >_n^{n+1}$$

$$\mathbf{r}_{p1}^{n+1/2} = \mathbf{r}^n + \Delta t/2 \mathbf{v}^n$$

$$\mathbf{E}, \mathbf{B}\left(\mathbf{r}_{\mathbf{p1}}^{\mathbf{n+1/2}}\right) = \sum_{ijk} \left(\mathbf{E}, \mathbf{B}_{ijk}\right) W\left(|\mathbf{r}_{ijk} - \mathbf{r}_{p1}^{n+1/2}|\right)$$

$$m_i \frac{d\mathbf{v}_{p1}^{n+1}}{dt} = e\left(\mathbf{v}^n \times + \mathbf{B}^{n+1/2} + \mathbf{E}^{n+1/2}\right)$$

$$N^{n+1} = \sum_p w_p \mathbf{v}_{p1}^{n+1} W\left(|\mathbf{r}_{ijk} - \mathbf{r}_{p1}^{n+1}|\right) \qquad u^{n+1} = \sum_p w_p \mathbf{v}_{p1}^{n+1} W\left(|\mathbf{r}_{ijk} - \mathbf{r}_{p1}^{n+1}|\right)$$

Prediction

$$\mathbf{B}_{p2}^{n+1} = \mathbf{B}^n - \Delta t \nabla \times \mathbf{E}^{n+1/2}$$

$$\mathbf{E}_{p2}^{n+1} = -\mathbf{u}^{n+1} \times \mathbf{B}_{p2}^{n+1} + \frac{\nabla \times \mathbf{B}_{p2}^{n+1}}{N^{n+1}} - \frac{\nabla \cdot \mathbf{P_e}}{N^{n+1}} + \eta \nabla \times \mathbf{B}_{p2}^{n+1} - \nu \nabla^2 \nabla \times \mathbf{B}_{\mathbf{p2}}^{\mathbf{n+1}}$$

$$\mathbf{r}_{p2}^{n+1/2} = \mathbf{r}^n + \Delta t/2 \mathbf{v}^n$$

$$(\mathbf{E}, \mathbf{B})^{n+1/2} =< (\mathbf{E}, \mathbf{B}) >_n^{n+1}$$

$$m_i \frac{d\mathbf{v}_{p2}^{n+1}}{dt} = e\left(\mathbf{v}^n \times + \mathbf{B}^{n+1/2} + \mathbf{E}^{n+1/2}\right)$$

$$N^{n+1} = \sum_p w_p \mathbf{v}_{p1}^{n+1} W\left(|\mathbf{r}_{ijk} - \mathbf{r}_{p1}^{n+1}|\right) \qquad u^{n+1} = \sum_p w_p \mathbf{v}_{p1}^{n+1} W\left(|\mathbf{r}_{ijk} - \mathbf{r}_{p1}^{n+1}|\right)$$

Correction

$$\mathbf{B}^{n+1} = \mathbf{B}^n - \Delta t \nabla \times \mathbf{E}^{n+1/2}$$

$$\mathbf{E}^{n+1} = -\mathbf{u}^{n+1} \times \mathbf{B}^{n+1} + \frac{\nabla \times \mathbf{B}^{n+1}}{N^{n+1}} - \frac{\nabla \cdot \mathbf{P_e}}{N^{n+1}} + \eta \nabla \times \mathbf{B}^{n+1} - \nu \nabla^2 \nabla \times \mathbf{B}^{\mathbf{n+1}}$$

$t + \Delta t$

Start writing this

```
while( time < final_time )
{
    predictor(state);
    predictor(state);
    corrector(state);
    diagnostics.dump(time);
    time += dt;
}
```

Then this

```
void predictor(State& state)
{
    faraday(state.B, state.E, Bpred);
    ampere(Bpred, state.J);
    ohm(Bpred, state.J, state.N, state.V, Epred);
    average(state.E, Epred, Eavg);
    average(state.B, Bpred, Bavg);
}
```

Only then this

C++

```
void faraday(Vector const& B, Vector const& E, Vector& Bnew)
{
    // Faraday's law: dB/dt = - curl(E)
    Bnew = B - dt * curl(E);
}
```

Functions

1. Small (1-20 lines max)
2. Do one thing (what its name says)
3. Use descriptive names (changes something? Getter? Runs something?…)
4. Prefer fewer arguments (0 best, 1 ok, 2 suspicious, 3 smells…)
5. Have no side effects

My code is clean, I write comments…

What are the problem with comments?

1. They're annoying to write
2. No one really reads them
3. They can unsync with code and be misleading when we do read them… so we don't read them…
4. They very often hide bad code that should be changed…

Comments are often procrastination to fix code

1. A block of code with a comment : is a FUNCTION
2. Write expressive code, not comments
3. Don't be redundant : « loops over elements… »

When to write comments?

1. Use as explanation of intent : why done this way? What assumptions?
2. Talks to colleagues or later you.

My code is clean, I write comments...

Bad comments

```
// loop over particles
for (auto const& particle : particles)
{
}
```

```
// return the size of the particle array
int particleArraySize() const
{
    return array_.size();
}
```

Useful comments

```
template<typename ResourcesView>
NO_DISCARD auto restart_patch_data_ids(ResourcesView const& view) const
{
    // true for now with https://github.com/PHAREHUB/PHARE/issues/664
    constexpr bool ALL_IDS = true;

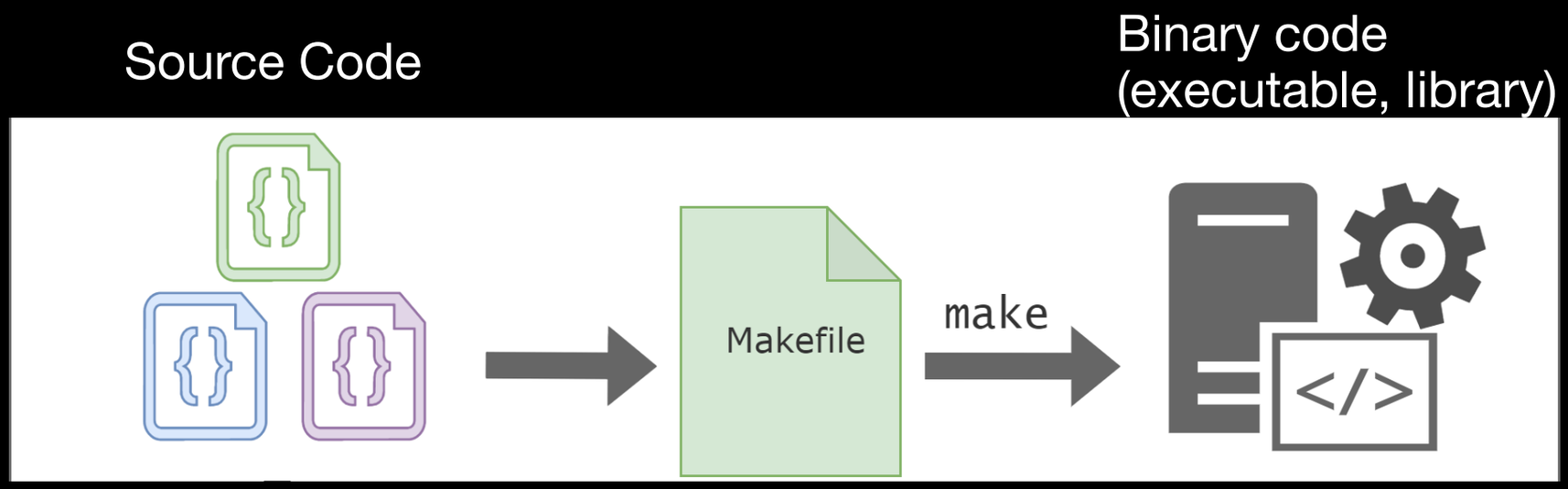
    std::vector<int> ids;
```

```
static void postprocessBy2d(auto& bx, auto& by, auto const& layout,
                            core::Point<int, dimension> idx)
{
    auto locIdx = layout.AMRToLocal(idx);
    auto ix     = locIdx[dirX];
    auto iy     = locIdx[dirY];
    //                              |
    //  here with offset = 0 -> -- -- <- or here with offset = 1
    //                              |
    if (idx[dirY] % 2 != 0)
    {
        int xoffset = (idx[dirX] % 2 == 0) ? 0 : 1;
        int yoffset = 1;

        by(ix, iy) = 0.5 * (by(ix, iy - 1) + by(ix, iy + 1))
                     + 0.25
                           * (bx(p_minus(ix, xoffset), d_minus(iy, yoffset))
                              - bx(p_plus(ix, xoffset), d_minus(iy, yoffset))
                              - bx(p_minus(ix, xoffset), d_plus(iy, yoffset))
                              + bx(p_plus(ix, xoffset), d_plus(iy, yoffset)));
    }
}
```


CMake

Writing Makefiles is difficult



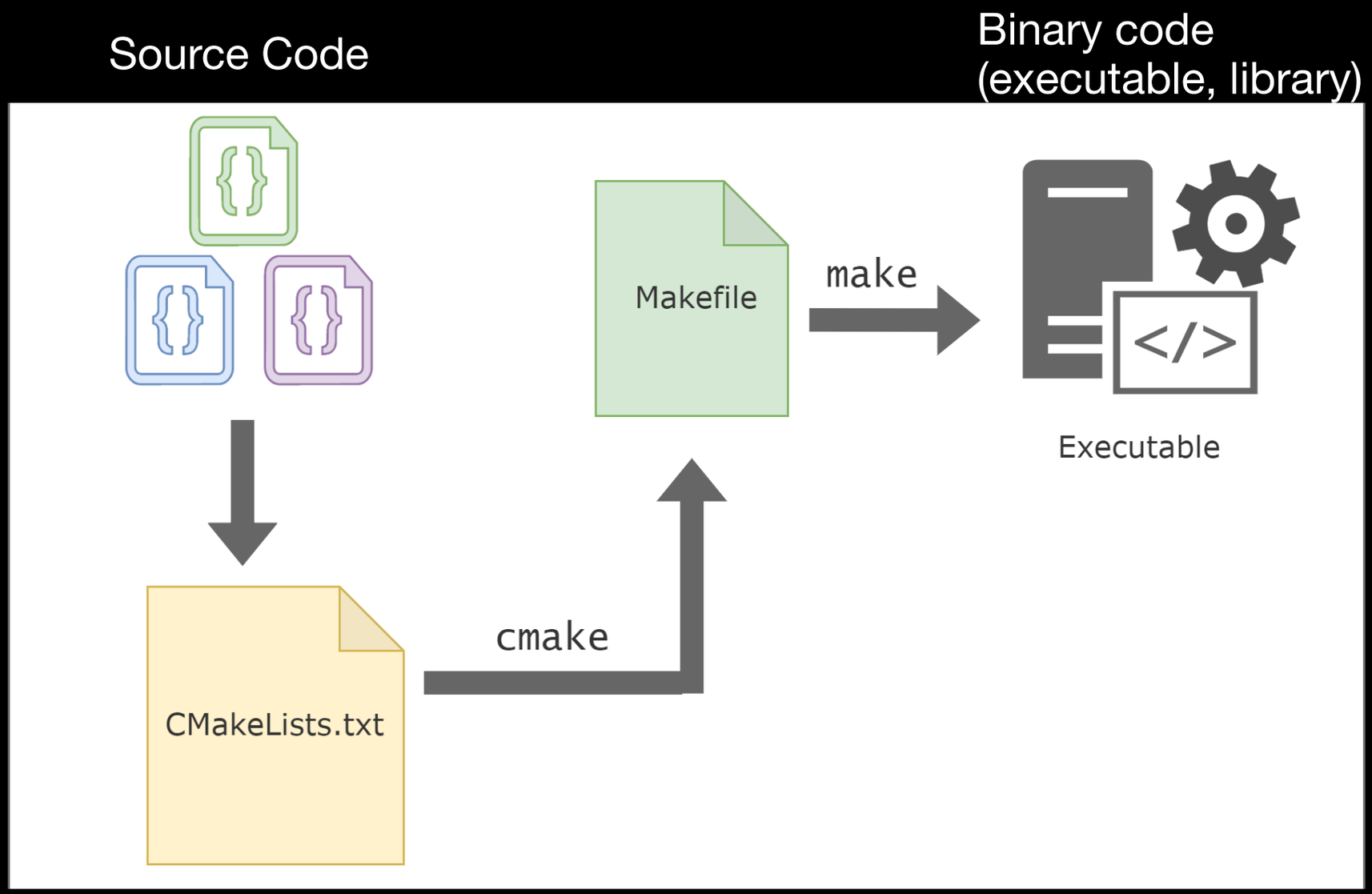
Makefile:

- compiler command for each compilation unit (sources, headers)
- Link command for assemble compiled units into targets (libraries, executables)

Writing makefiles can become very complicated:

- When creating multiple targets
- Each target has different dependencies
- Finding dependencies and dealing with their versions
- Build portability across different platforms

CMake writes the Makefile for you



CMakeLists.txt: high level description of each target and their dependencies

Make reads CMakeLists.txt files and generate the Makefile

CMake

- Made by ‘kitware’, open source
- Available at https://cmake.org/
- Packaged on Linux/Mac (e.g. homebrew)
- Tutorial: https://cmake.org/cmake/help/latest/guide/tutorial/index.html

People love to hate CMake, but it is widely used…

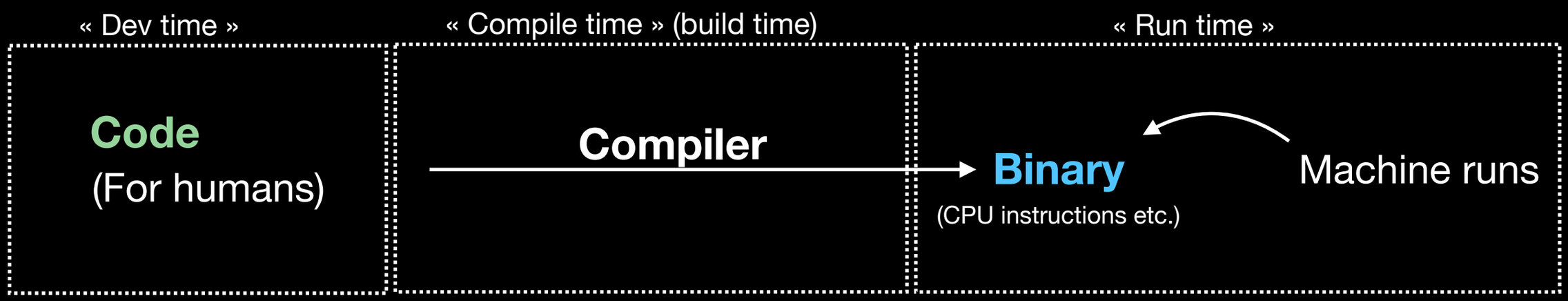
MESON: a good alternative

https://mesonbuild.com





Interpreted vs Compiled language





Interpreting code is slower than directly running binary CPU instructions
Compiled code is more optimized
Interpreted languages are more flexible

HPC?

- The goal of HPC is to execute a program « *as fast as possible* »
- HPC codes are thus written mostly with compiled languages
- « modern » codes mix compiled languages for (heavy) computational components and interpreted code for higher level or user interface components.

Main HPC compiled languages

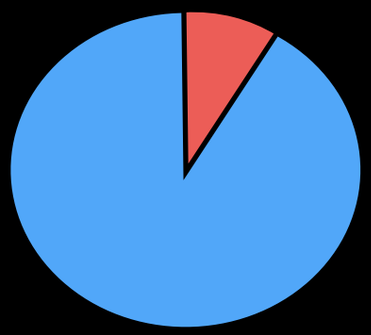
- FORTRAN (FORmula TRANslation): 1957 and many versions since, latest is Fortran 2023
  - « easy » to write code that « looks like the math »
- C: invented in 1972 - general **low level** (fast) programming language
- C++: invented in 1985 - low level of C with easier high level abstraction

Main HPC interpreted languages

- Python: invented in 1991, really took off for science in mid 2000s (Numpy 2006, etc.)
- Huge ecosystem, easy to write

Evolution of HPC codes

< 2000-2010

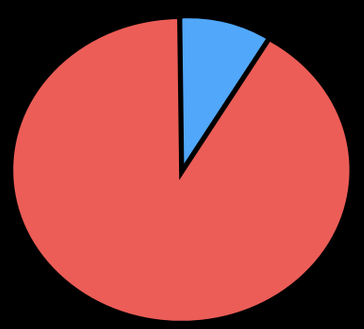


Soft. Engineering

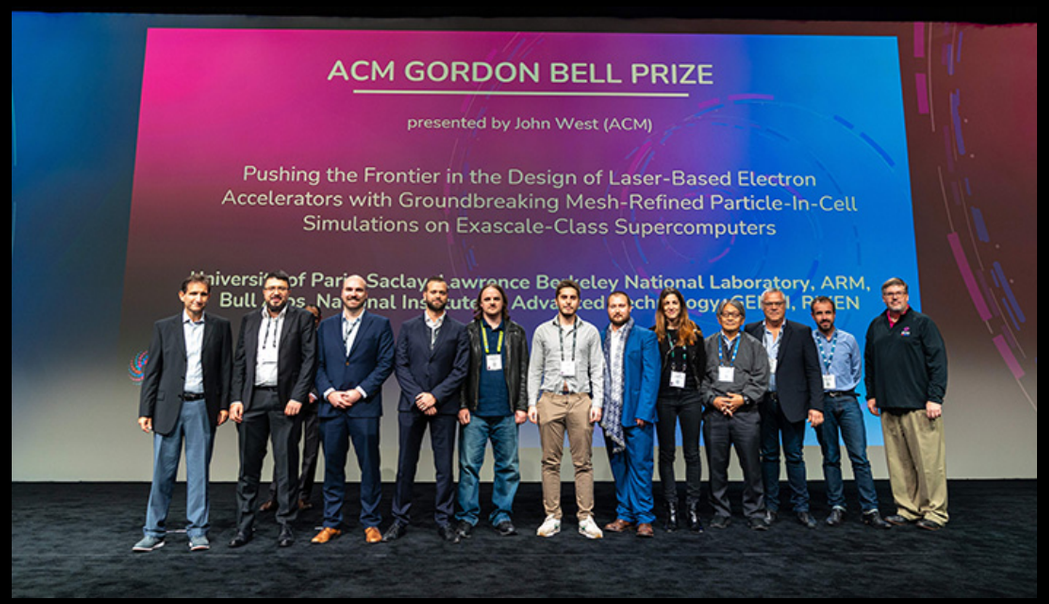
vs

numerical

> 2000-2010







- Code is mostly solving equations
- Simple supercomputer architectures
- 1 dev, 1 user, usually same person (no versatility, no ergonomic interface, etc.)

- Peta (1e15) /Exascale (1e18) scalability
- Complex data structures
- Complex architectures (multi-GPU / CPUs etc.)
- Versatility, user communities

Why C++?

We (the HPC community) are outnumbered by industry people

- We run on computers made for games (or now AI)
- Programming languages (and interpreters/compilers) are driven by industry needs
- Third party libraries are mostly developed and maintained for industry needs
- Need to hire people with engineering and hardware/low-level skills: those people typically are in the industry

What does the industry?

(TIOBE index)

Jul 2025	Jul 2024	Change	Programming Language	Ratings	Change
1	1		Python	26.98%	+10.85%
2	2		C++	9.80%	-0.53%
3	3		C	9.65%	+0.16%
4	4		Java	8.76%	+0.17%
5	5		C#	4.87%	-1.85%
6	6		JavaScript	3.36%	-0.43%
7	7		Go	2.04%	-0.14%
8	8		Visual Basic	1.94%	-0.13%
9	24	⇈	Ada	1.77%	+0.99%
10	11	↑	Delphi/Object Pascal	1.77%	-0.12%
11	30	⇈	Perl	1.76%	+1.10%
12	9	↓	Fortran	1.67%	-0.38%

Why C++?





python

Performance

Ergonomy

Your bible: https://cppreference.com/

Online compilers (for quick tests): https://coliru.stacked-crooked.com/ (quick & dirty cpp tests)

https://godbolt.org/ (See assembly etc.)

Some references:

https://nicolasaunai.github.io/teaching/M2IRT-HPC-Master-class

Headers: definitions of classes, functions, etc. that are used after

Function name

argument

Mandatory main function (point of entry)

```
1
2  #include <iostream>
3  #include <string>
4
5  int my_function(std::string word)
6  {
7      return word.size();
8  }
9
10
11 int main()
12 {
13     std::string hello = "hello world";
14
15     std::cout << hello <<" is "<< my_function(hello) << " letters long\n";
16 }
17
```

Basics: https://coliru.stacked-crooked.com/a/d4a3e9d19986f3e7

C++ IN A NUTSHELL

- Data types
- Pointers & references: https://coliru.stacked-crooked.com/a/4c8c292d1297b138
- Basics https://coliru.stacked-crooked.com/a/c99925c0e0f312dc:
  - Main function,
  - Functions, return type, arguments, default arguments, lambda functions
- Structures & classes: https://coliru.stacked-crooked.com/a/8d89c5d1ae4c2f57
  - public, private,
  - Constructors, destructors, methods, operators
  - Private inheritance: sharing code
  - Public inheritance, polymorphism, Liskov principle, design patterns
- Containers (array, vector…): https://coliru.stacked-crooked.com/a/99792e2d07efd28b
- Smart pointers & Ownership
- STL (transform, random, math, etc.)
- Templates, variadic templates
- Exceptions
- Const, constexpr
- namespace

Some general advices…

- **const** by default, and remove it if required : help the compiler
- **constexpr**, don’t make at runtime if you can make it at build time
- **Avoid virtual** calls or if at **low level**
- Keep memory **allocation out of your heavy loops**
- **Beware copies**… start by deleting copy constructors…

HANDS ON!



SETUP

Install opencode

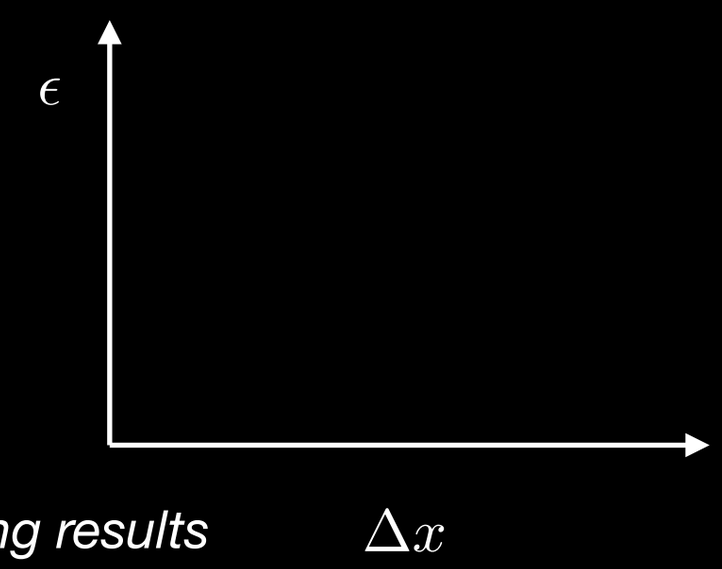
Make a GitHub account

Make a new Cmake C++ project where you :

- Demonstrate these finite difference formulae for the first order derivative are respectively first and second order accurate

$$\frac{f_{i+1} - f_i}{\Delta x} = f'(x)$$

$$\frac{f_{i+1} - f_{i-1}}{2\Delta x} = f'(x)$$



- *I want results written in hdf5 files*
- *And a python Jupyter notebook explaining and showing results*