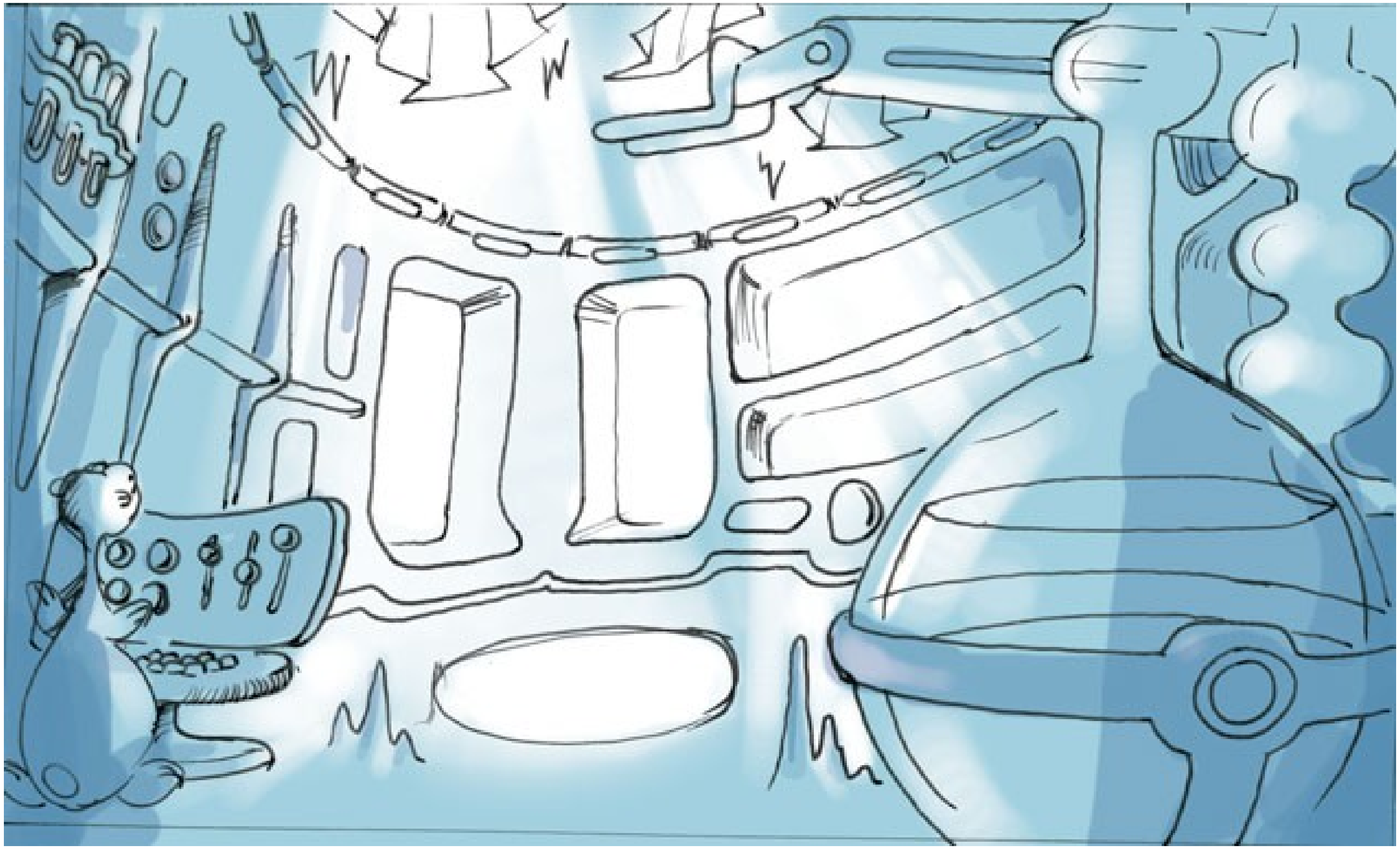
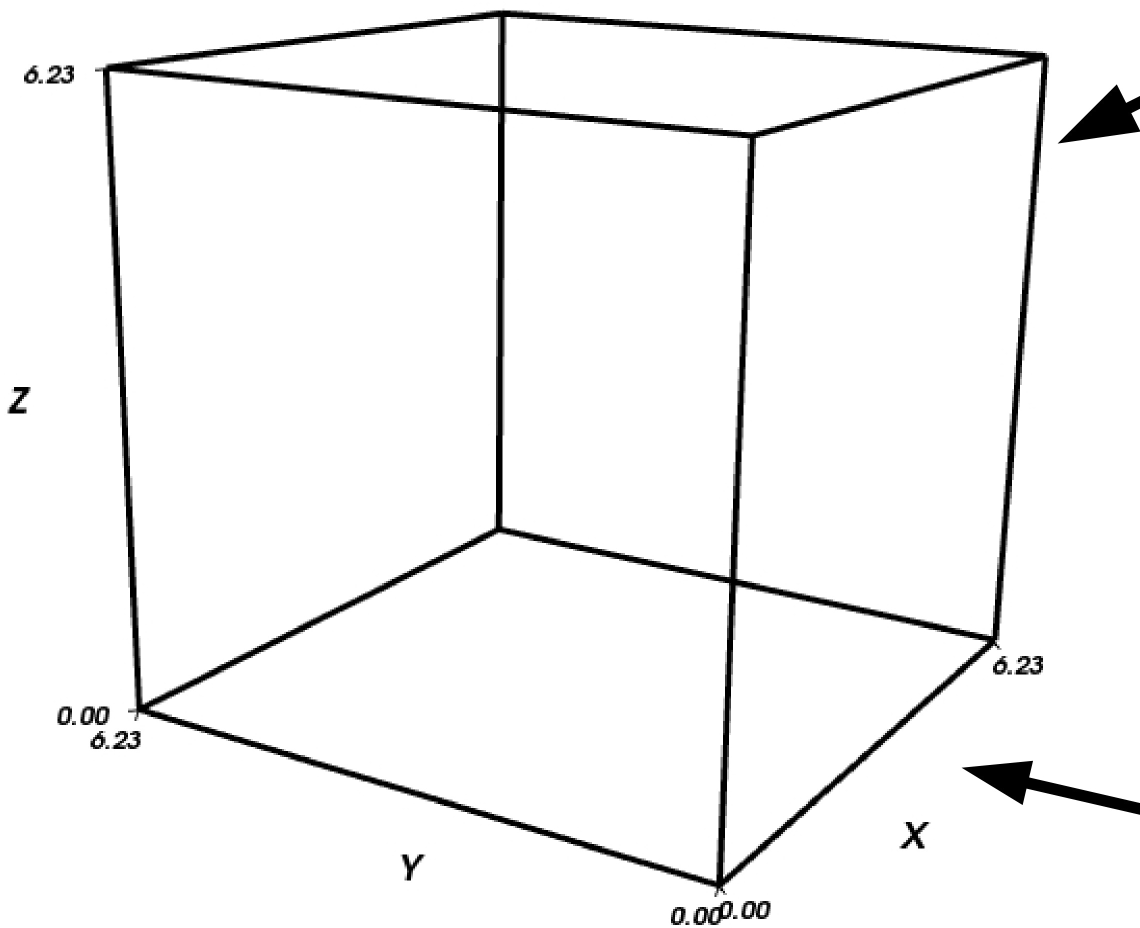
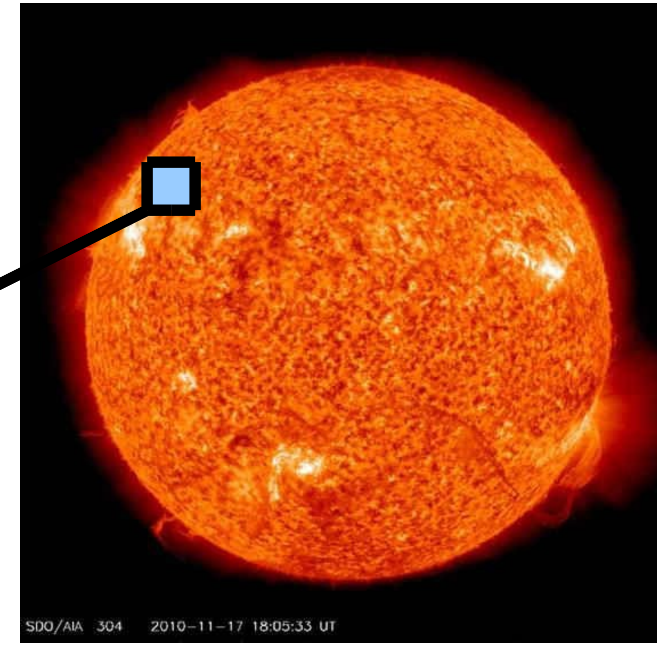
Periodic box simulations and tools

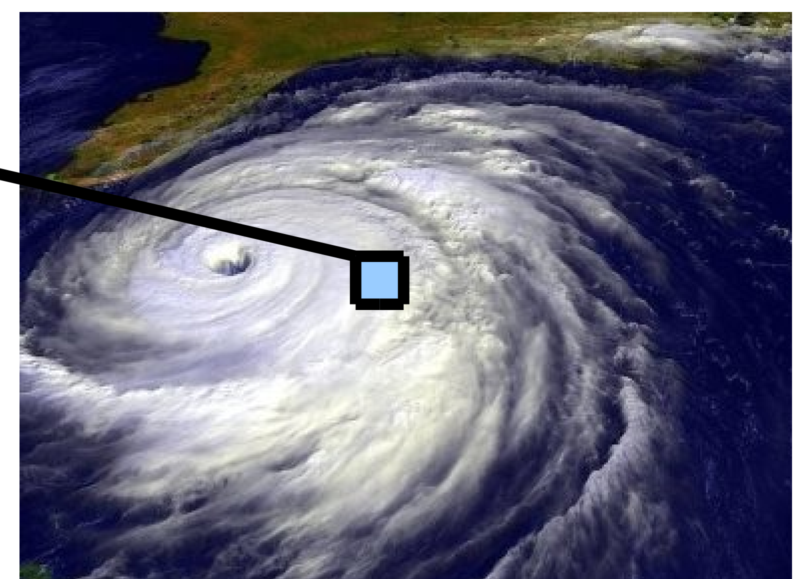
Yannick Ponty
Laboratoire Lagrange CNRS
Observatoire de la Côte d'Azur
University of the Côte d'Azur (UCA)

Do we need complex geometry ?

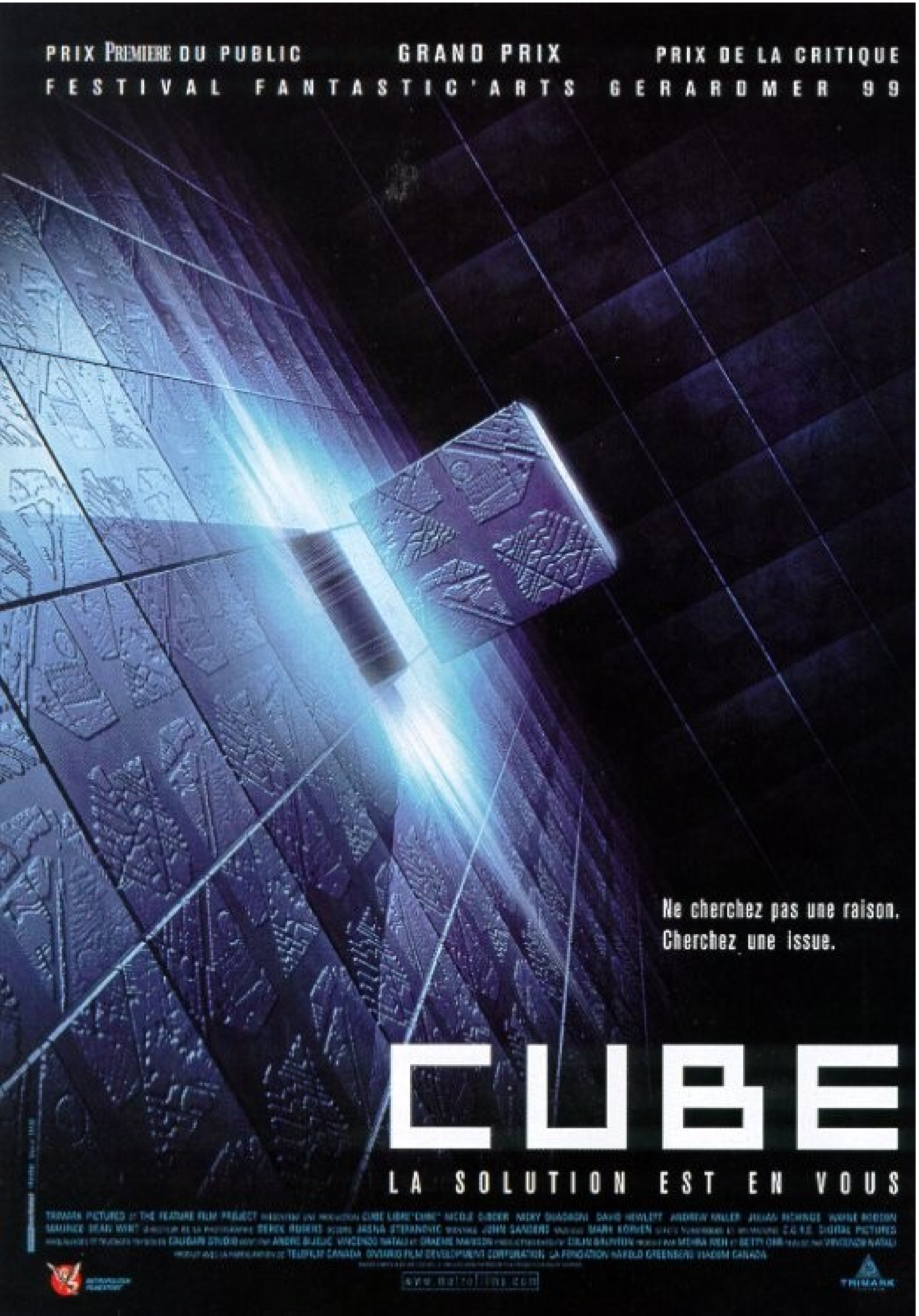


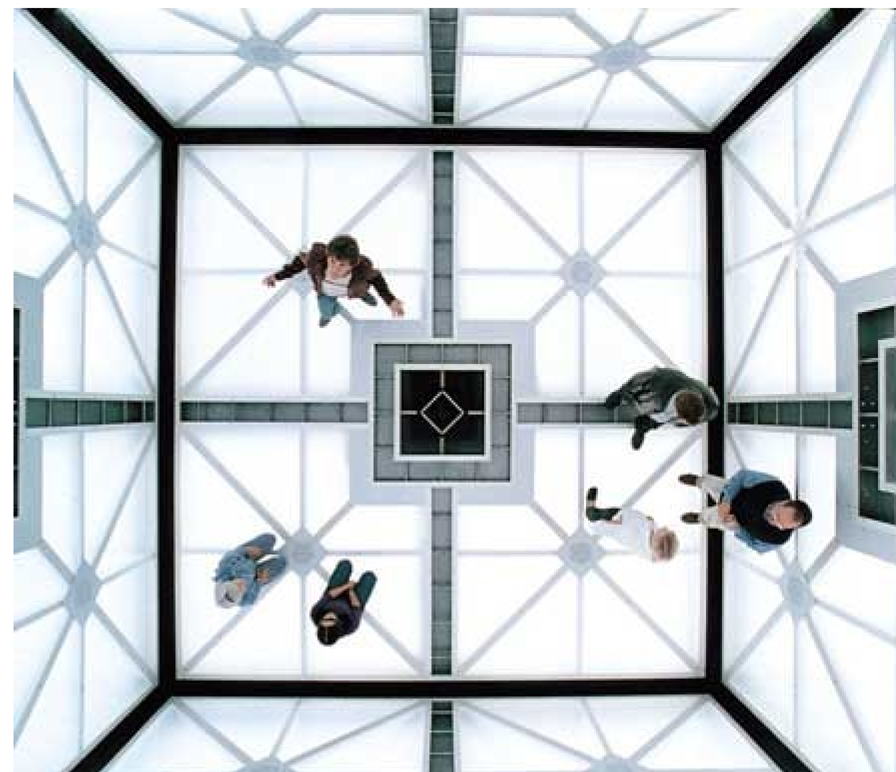






$(2\pi)^3$ ***periodic box***


PRIX PREMIERE DU PUBLIC
GRAND PRIX
PRIX DE LA CRITIQUE
FESTIVAL FANTASTIC'ARTS GERARDMER 99
Ne cherchez pas une raison.
Cherchez une issue.
CUBE
LA SOLUTION EST EN VOUS



Let's go inside the

Outline

Numerical method

(How make a cube yourself : The unrevealed story !)

1) The periodic box a numerical experiment

Fundamental equation, Basic non dimensional number

Pressure, projector , Numerical schema

Parallelization procedure, Different forcing

tracers

2) Numerical quantities output and experimental observable

Probe, time signal

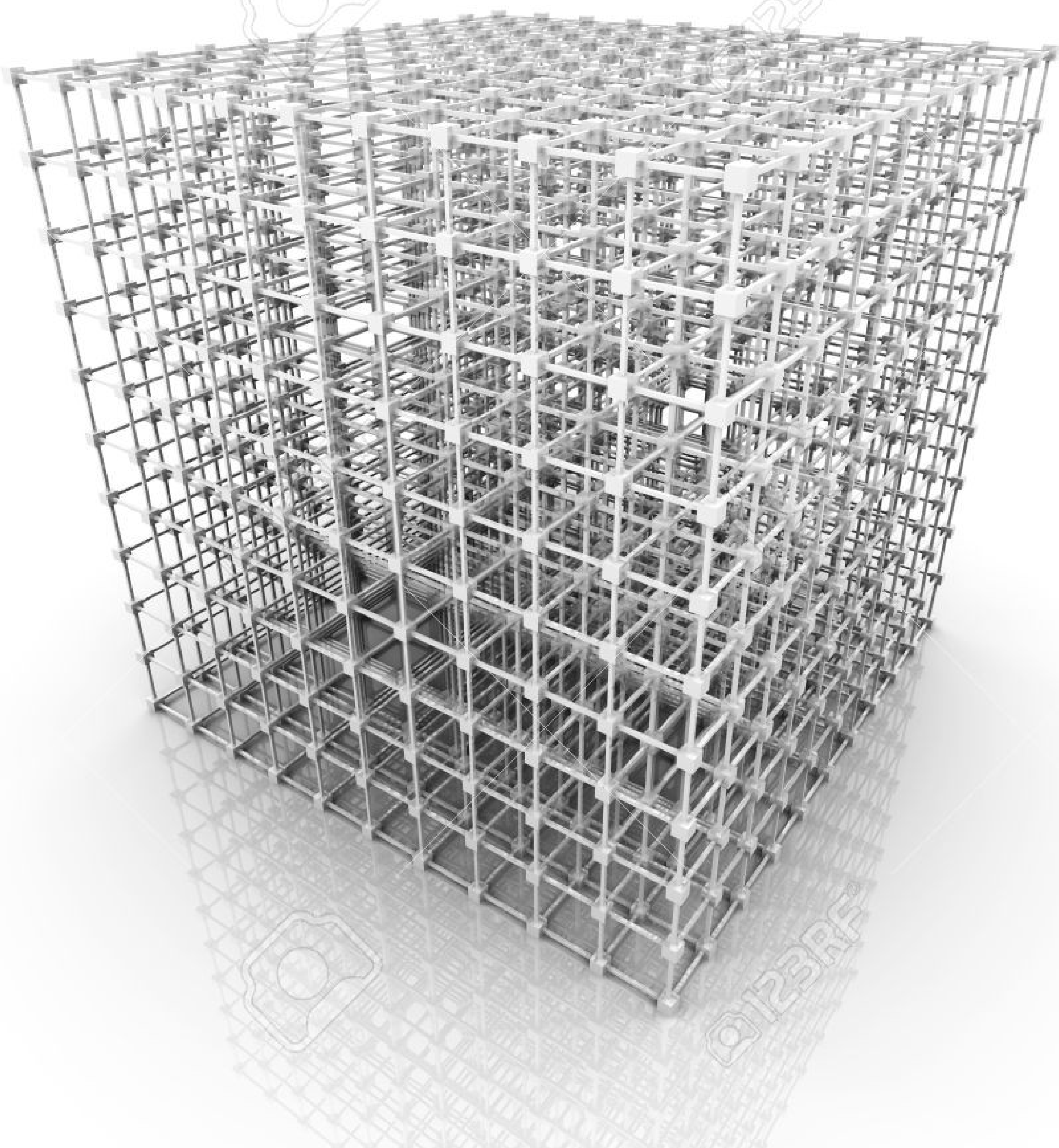
Mean flow

3D visualization

3) LES, energy transfer

4) Penalization method

Regular Computing grid



incompressible MHD equations in a periodic box

$$\partial_t \vec{u} + \vec{u} \nabla \vec{u} = -\nabla P + \nu \Delta \vec{u} + \vec{j} \times \vec{b} + \vec{F}$$

$$\partial_t \vec{b} = \nabla \times (\vec{u} \times \vec{b}) + \eta \Delta \vec{b}$$

$$\nabla \vec{u} = 0 \quad \nabla \vec{b} = 0$$

$$\vec{u}(x,y,z,t) = \vec{u}(x+2\pi i, y+2\pi j, z+2\pi k, t)$$

$$\vec{b}(x,y,z,t) = \vec{b}(x+2\pi i, y+2\pi j, z+2\pi k, t)$$

$$i,j,k = 0,1,2,\ldots\infty$$

Pseudo-spectral method

Real space

spectral space

$$\vec{u}(\vec{x},t) \xleftrightarrow{\text{FFT}} \hat{u}(\vec{k},t)$$

Fast Fourier Transform

$$\partial_x \vec{u}(\vec{x},t) \longleftarrow i k_x \hat{u}(\vec{k},t)$$

$$w(\vec{x},t) = \nabla \times \vec{u}(\vec{x},t) \longleftarrow i\vec{k} \times \hat{u}(\vec{k},t)$$

$$h(\vec{x},t) = \vec{u}(\vec{x},t) \times \vec{w}(\vec{x},t) \longrightarrow \hat{h}(\vec{k},t)$$

Numerical convergence

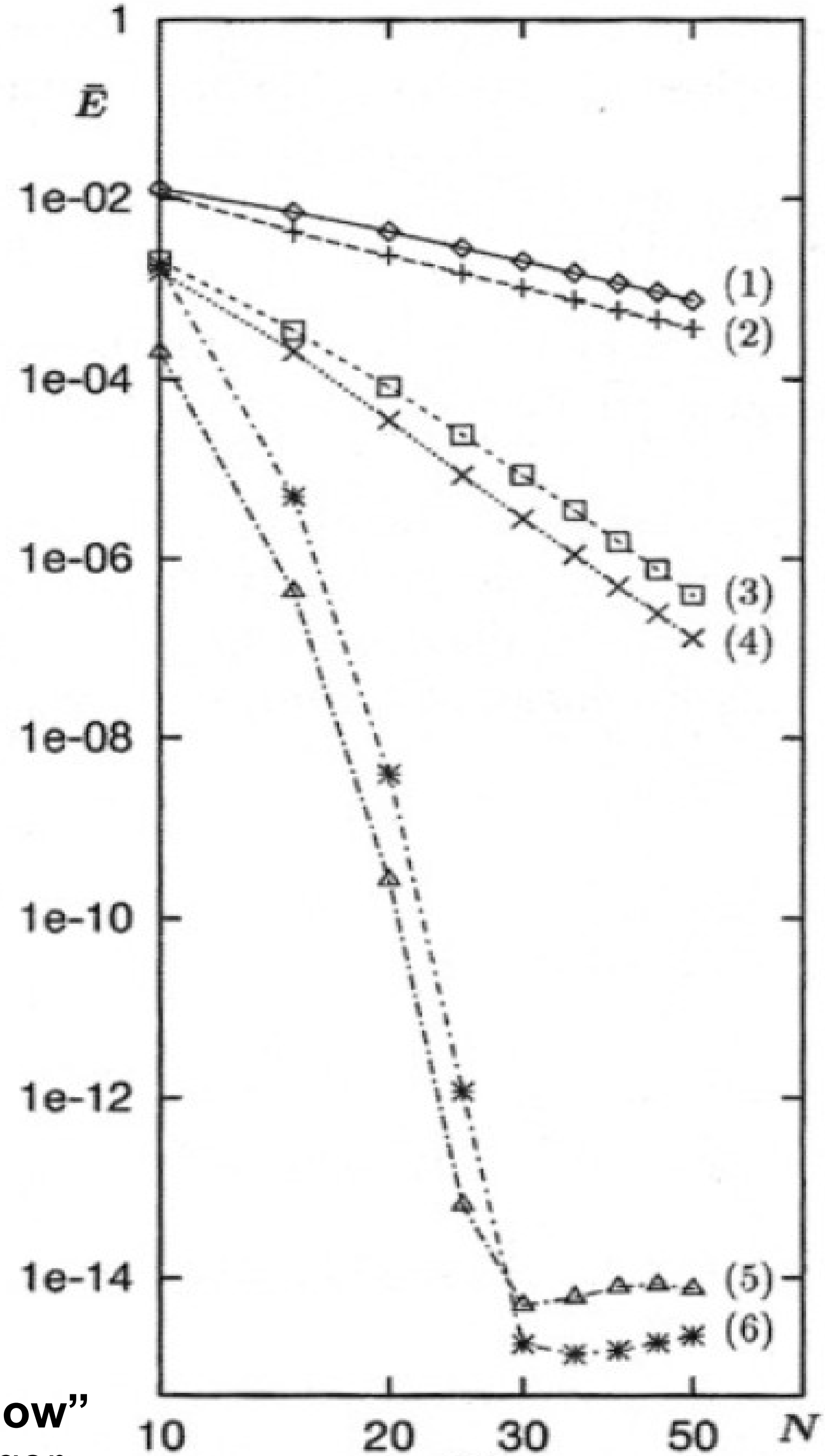
Helmotz equation

$$-\nu u'' + u = f, \quad -1 < x < 1, \quad \nu = 10^{-2}$$

$$u(-1) = g_-, \quad u(1) = g_+,$$

$$u(x) = 1 - \frac{\sinh\left[(x+1)/\sqrt{\nu}\right]}{\sinh\left(2/\sqrt{\nu}\right)}$$

(1) Second-order finite-difference method with uniform mesh ($\Delta x = 2/N$).
(2) Second-order finite-difference method with Gauss-Lobatto mesh (3.75).
(3) Sixth-order Hermitian method with uniform mesh ($\Delta x = 2/N$).
(4) Sixth-order Hermitian method with Gauss-Lobatto mesh (3.75).
(5) Chebyshev collocation method [Gauss-Lobatto mesh (3.75].
(6) Chebyshev tau method (polynomial degree = N).



(a)

"Spectral methods for incompressible Viscous Flow"
R. Peyret *Applied Mathematical Sciences 148 Springer.*

Numerical method :

$$\partial_t \vec{u} + \vec{u}\nabla\vec{u} = -\nabla P + \nu\Delta\vec{u} + \vec{j}\times\vec{b} + \vec{F}$$

$$\partial_t \vec{u} - \nu\Delta\vec{u} = \vec{u}\times\vec{\omega} - \nabla\left(P + \frac{\vec{u}^2}{2}\right) + \vec{j}\times\vec{b} + \vec{F}$$

$$\vec{\omega} = \nabla\times\vec{u}$$

Elimination of the pressure :

Projector Π in the solenoidal space function :

In spectral space the projector tensor : $\Pi_{ij} = \delta_{ij} - \frac{k_i k_j}{k^2}$

$$\partial_t \vec{u} - \nu\Delta\vec{u} = \Pi\left[\vec{u}\times\vec{\omega} + \vec{j}\times\vec{b} + \vec{F}\right]$$

Numerical method :

$$\partial_t \hat{u} - \nu k^2 \hat{u} = \Pi [\widehat{\vec{u}\times\vec{\omega}} + \widehat{j\times\vec{b}} + \hat{F}]$$

$$\partial_t \hat{b} - \eta k^2 \hat{b} = \widehat{\nabla} \times (\widehat{\vec{u}\times\vec{b}})$$

The diffusion term is implemented implicitly with the exponentiel method.

$$\partial_t U_k(t) = -\nu k^2 U_k(t) + G_k(t)$$

$$\partial_t (e^{\nu k^2 t} U_k) = e^{\nu k^2 t} G_k(t)$$

For the other time step the forward Euler - Adams-Basford shemes is implemented :

$$\frac{e^{\nu k^2 (t+\Delta t)} U_k(t+\Delta t) - e^{\nu k^2 t} U_k(t)}{\Delta t} = \frac{3}{2} G_k(t) e^{\nu k^2 t} - \frac{1}{2} G_k(t-\Delta t) e^{\nu k^2 (t-\Delta t)}$$

$$U_k(t+\Delta t) = U_k(t) e^{-\nu k^2 \Delta t} + e^{-\nu k^2 \Delta t} \Delta t \left[\frac{3}{2} G_k(t) - \frac{1}{2} G_k(t-\Delta t) e^{-\nu k^2 \Delta t}\right]$$

"Spectral methods for incompressible Viscous Flow" R. Peyret
Applied Mathematical Sciences 148 Springer.

Aliasing removal :

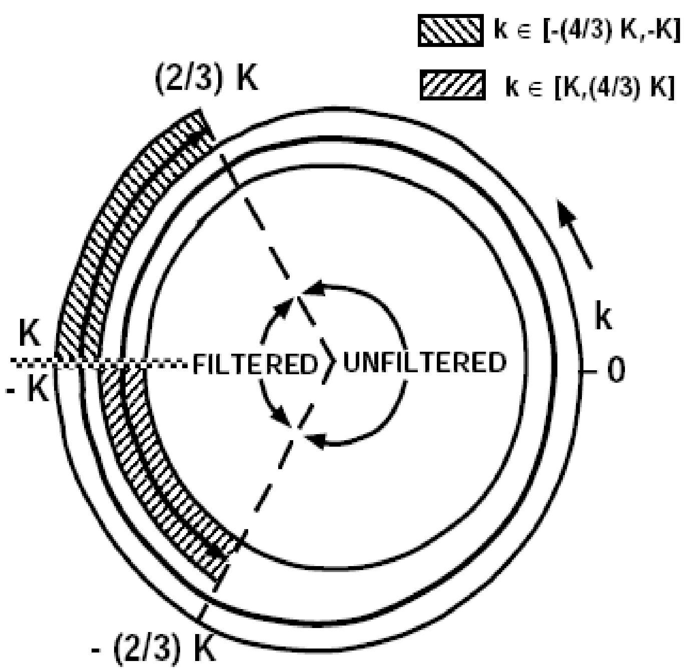
Aliasing can cause numerical instability in the time integration of *nonlinear* equations. For example, a typical quadratically nonlinear term is

$$u\,u_x = \left(\sum_{p=-K}^{K} a_p\, e^{ipx}\right)\left(\sum_{q=-K}^{K} i\,q\,a_q\, e^{iqx}\right) \qquad (11.10)$$

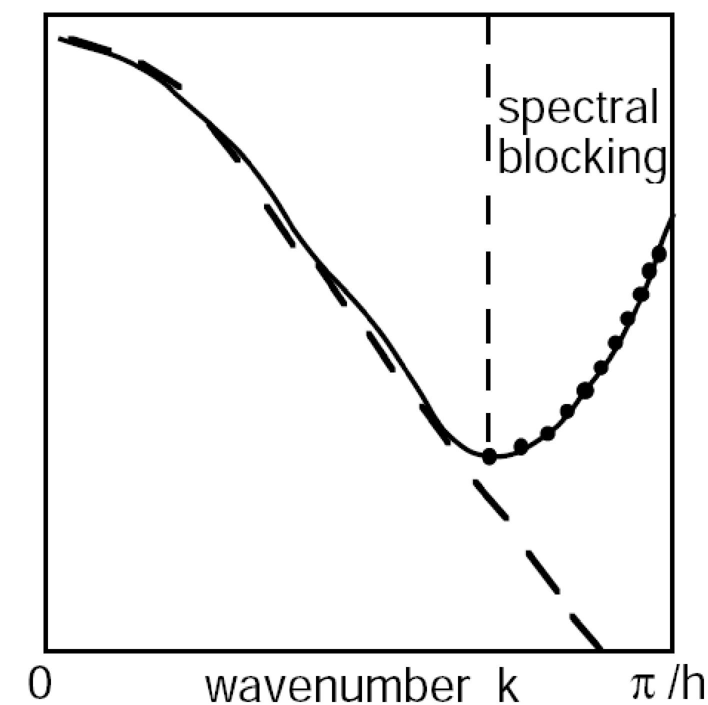
$$= \sum_{k=-2K}^{2K} b_k\, e^{ikx} \qquad (11.11)$$

where the b_k are given by a sum over products of the a_k. The nonlinear interaction has generated high zonal wavenumbers which will be aliased into wavenumbers on the range $k \in [-K, K]$, creating a wholly unphysical cascade of energy from high wavenumbers to low.

DEALIASING AND THE ORSZAG TWO-THIRDS RULE



$k > 2/3 K_{max}$



Forcing :

$$\partial_t \vec{u} + \vec{u} \nabla \vec{u} = -\nabla P + \nu \Delta \vec{u} + \vec{F}$$

Energy Injection :

Constant force (Torque) $\vec{F} = \vec{F}(x,y,z)$ $\quad \varepsilon_i = \vec{F} . \vec{u}$

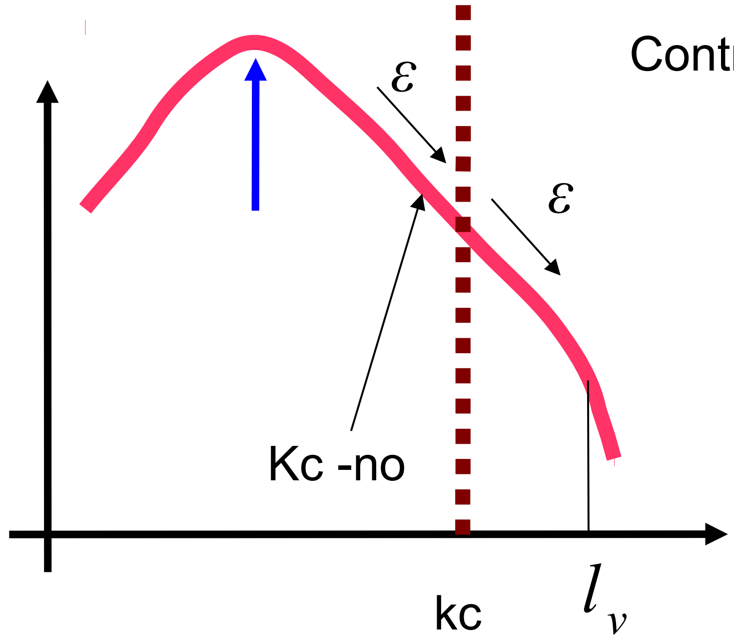
Add + F a each time step

Constant velocity : $\vec{u}(\vec{k}_i) = \vec{u}_F$ $\quad \varepsilon_i = fft^{-1}(\hat{u}(k_i) - \hat{u}_F(k_i)) . \vec{u}$

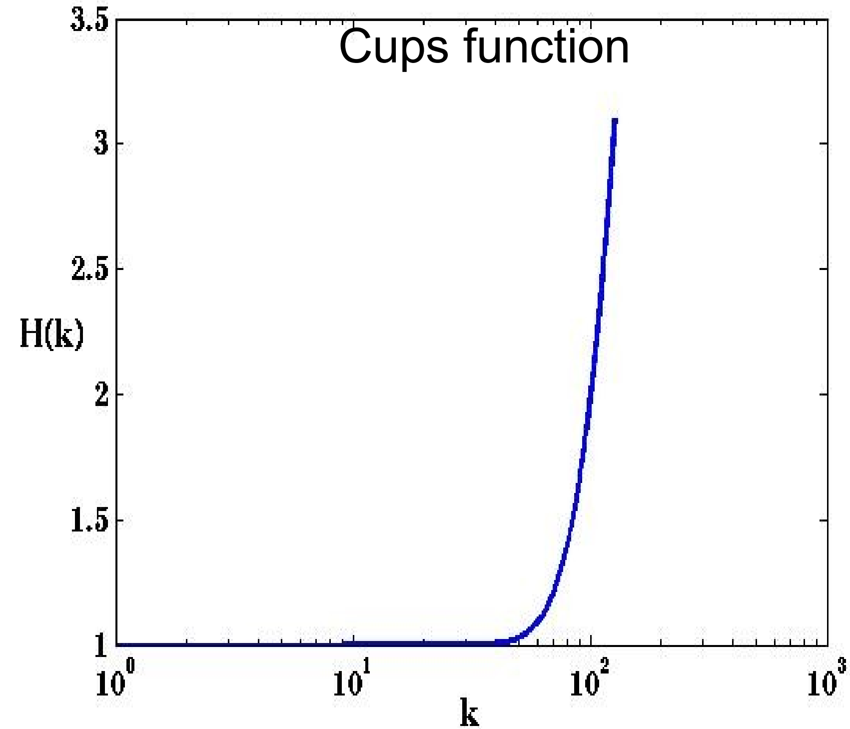
Keep some spectral modes constants at each time step

Constant energy

Large eddy simulation (LES) : J.P Chollet & M. Lesieur J. Admos. Sci **38** (1981)



Control the transfer into the subgrid scale



$$\nu(k,t) = 0.441\, C_k^{\frac{3}{2}} \sqrt{\frac{E(K_c - n_o, t)}{K_c - n_o}}\, H(k)$$

Ref: **"Turbulence in Fluids"** M. Lesieur *Kluwer*

"Large Eddy Simulation for incompressible flow" P. Sagaut *Springer*

Non dimensional Parameters

Reynolds number :

$$Rv = \frac{LU}{\nu}$$

Magnetic Reynolds number :

$$Rm = \frac{LU}{\eta} = \frac{L/U}{L^2/\eta}$$

eddy turn over time

magnetic diffusion time

Magnetic Prandtl number:

$$Pm = \frac{\nu}{\eta} = \frac{Rm}{Rv} = \frac{\tau_\eta}{\tau_\nu}$$

magnetic diffusion time

viscous time scale

Liquid Metal :

$$Pm \sim 10^{-5}$$
$$Rm \sim 100$$

→

$$Rv \sim 10^7$$

How to choose those quantities : **U , L** ?

$$\partial_t \vec{u} + \vec{u} \nabla \vec{u} = -\nabla P + \nu \Delta \vec{u} + \vec{j} \times \vec{b} + \vec{F}$$

$$\nabla \vec{v} = 0$$

U ~ 1 , F ~ 1 Order one

$$Rv = \frac{LU}{\nu}$$

A posteriori :

L = size of the box

$$U_{rms} = \sqrt{2 E_V}$$

$$L_i = \frac{\sum E(k)/k}{\sum E(k)}$$

$$u_{max}$$

Grashof number $$Gr = \frac{FL^3}{\nu^2}$$

Turbulence and mesh point :

$$R_v = \frac{L_i U_{rms}}{\nu}$$

64^3	TG1	R ~ 200
128^3	TG1	R ~ 460
256^3	TG1	R ~ 900
512^3	ABC1	R ~ 3500

Y. Ponty et al 2007 (Dynamo, Mhd)

Run	N	f	k_F	ν	R_e
I	256	TG	2	2×10^{-3}	675
II	512	TG	2	1.5×10^{-3}	875
III	1024	TG	2	3×10^{-4}	3950
IV	256	ABC	10	2.5×10^{-3}	275
V	256	ABC	3	2×10^{-3}	820
VI	512	ABC	3	6.2×10^{-4}	2520
VII	1024	ABC	3	2.5×10^{-4}	6200
VIII	256	RND	1	1.5×10^{-3}	2030

P. Mininni et al 2006 (Hydro)

$$K_{max}\,\eta \sim 1$$

512^3	R ~ 2040
1024^3	R ~ 7700
2048^3	R ~ 18 000
4096^3	R ~ 45 000

$$K_{max}\,\eta \sim 2$$

512^3	R ~ 1150
1024^3	R ~ 2666
2048^3	R ~ 6100
4096^3	R ~ 15 000

Kaneda et al *PoF* 15 2003 (hydro)
(Earth Simulator, japan)

Let's stir of cup of tea

Radius of the cup R = 0.05 m (5 cm)

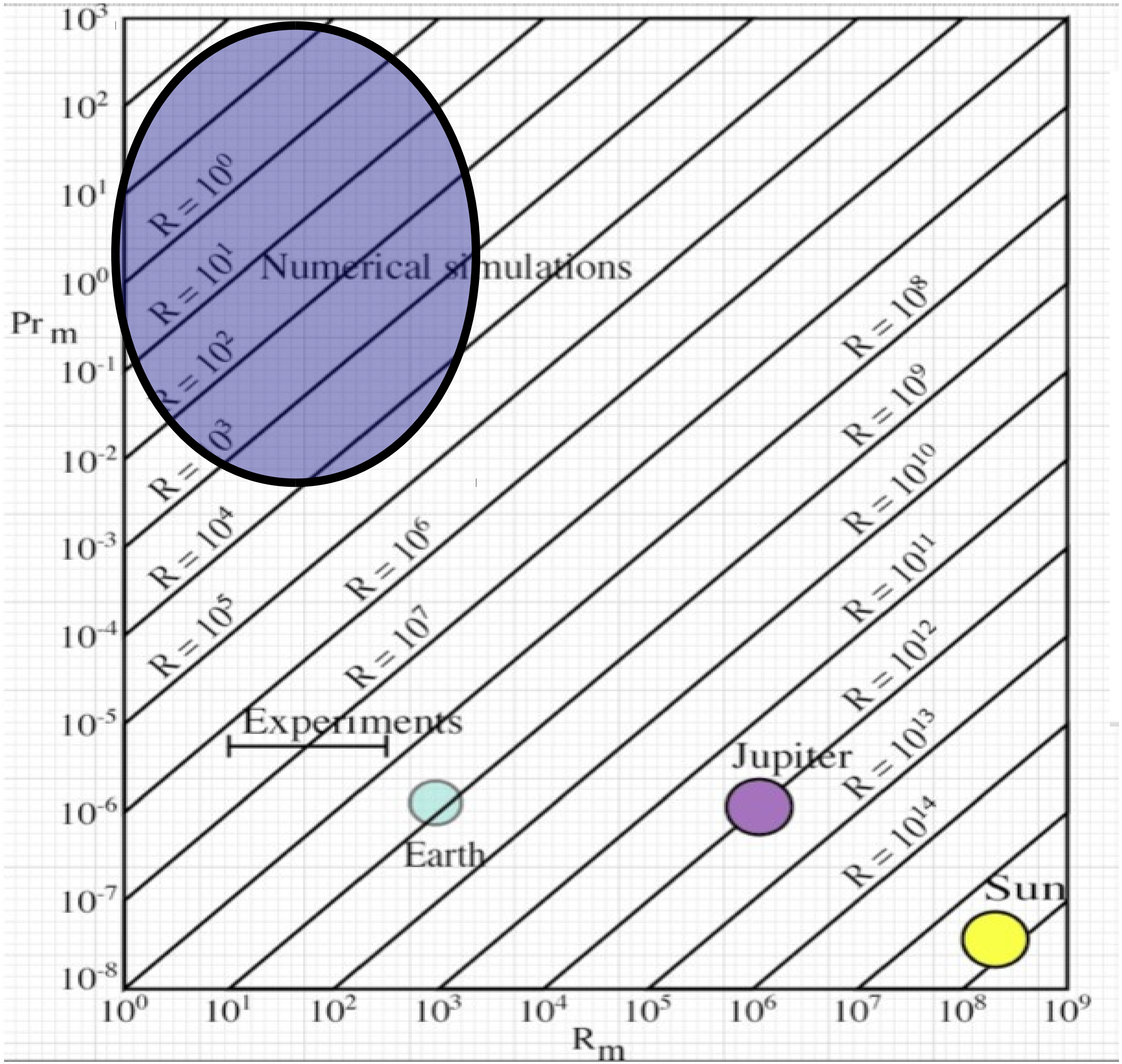
Motion of the spoon : 1 turn by second
V = 2 pi * R / 1 s = 0.01570 m/s

$$R_v = \frac{2\, pi\, R^2}{\nu\, \tau}$$

Kinematic viscosity of water :

5 °C	**1.52 10^(-6) m^2/s**	**R ~ 10 000**	
20 °C	**1.0 10^(-6) m^2/s**	**R ~ 15 000**	4096³ R ~ 15 000
30 °C	**0.804 10^(-6) m^2/s**	**R ~ 19 000**	

Proposal : Exascale computing to solve
HOT or a larger cup of tea turbulence



Numerical simulations
Experiments
Earth
Jupiter
Sun
$R = 10^0$
$R = 10^1$
$R = 10^2$
$R = 10^3$
$R = 10^4$
$R = 10^5$
$R = 10^6$
$R = 10^7$
$R = 10^8$
$R = 10^9$
$R = 10^{10}$
$R = 10^{11}$
$R = 10^{12}$
$R = 10^{13}$
$R = 10^{14}$
Pr_m
R_m

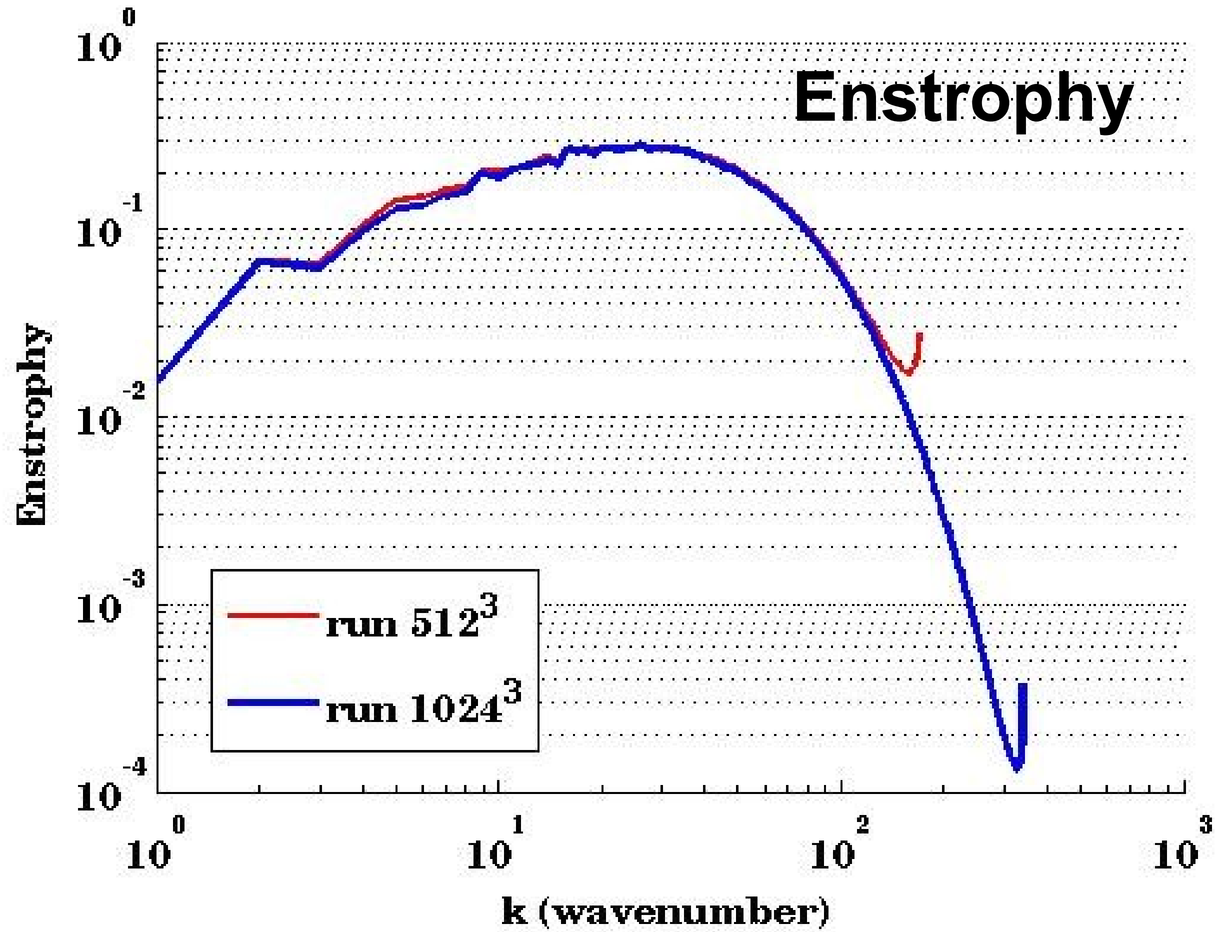
Grid Resolution !!!

$$\partial_t E = I - 2\nu\Omega$$

$$\eta_{hydro} \sim \left(\frac{\nu^3}{<\varepsilon>}\right)^{1/4} \doteq \left(\frac{2\Omega}{\nu^2}\right)^{-1/4}$$

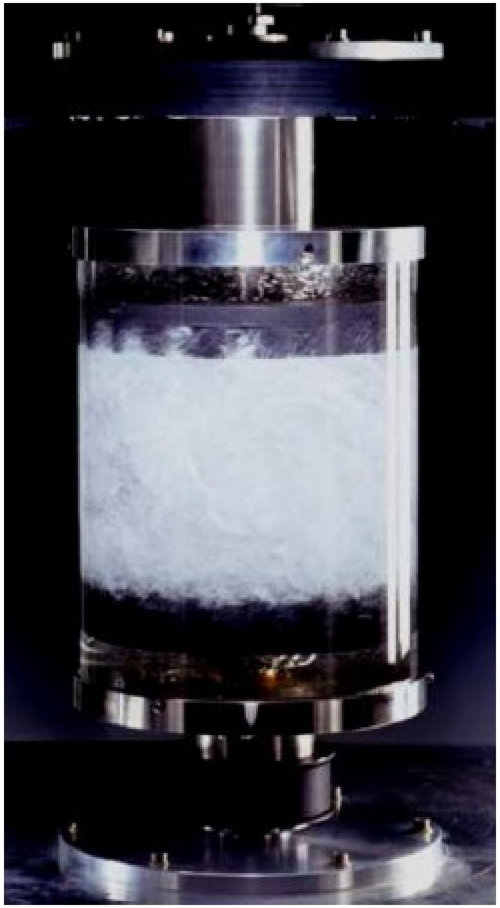
$$k_{max-deliasing}.\eta_{hydro} \sim 1 \text{ ou} \sim 2.$$

$$2/3k_{max}\left(\frac{2\Omega}{\nu^2}\right)^{-1/4} \sim 2$$



512^3 $2/3k_{max}\,\eta_{hydro} \sim 1.21$

1024^3 $2/3k_{max}\,\eta_{hydro} \sim 2.41.$



DNS of Von Karman Turbulent

$$L_0 \sim 1\,m$$
$$R_e \sim 10^6$$
$$\longrightarrow \quad \frac{L_0}{\eta} \sim R_e^{3/4} \sim 32\,000$$

Need at least $65536^3 = 2^{16}$

Need 20 scalar fields in double precision

$$65536^3 * 20 * 8\, octets = 40\ peta\, octets$$

40 Po / 10,649,600 cores ~ 3.6 Go by core !

1St rank TOP500 : Sunway TaihuLight CHINA

1310720 GB = 1,3 Po > 40 Po !!!

Computer time and mesh size : (double precision)

64^3	dt=0.002	1 time step : 0.72 s mono-proc
128^3	dt=0.001	1 time step : 7.4 s mono-proc
256^3	dt=0.0005	1 time step : 58.54 s mono-proc
512^3	dt=0.0003	1 time step : 464 s mono-proc

CFL condition : $dt < \frac{1}{U_{max} K_{max}}$ -> Factor 16 at least

100 over time: 64^3 -> 10 h mono proc
128^3 -> 205 h mono proc = 8 d 13 h
256^3 -> 3252 h mono proc = 135 d
512^3 -> ~ 5 y mono proc = 167 h ~ 7 d with 256 proc

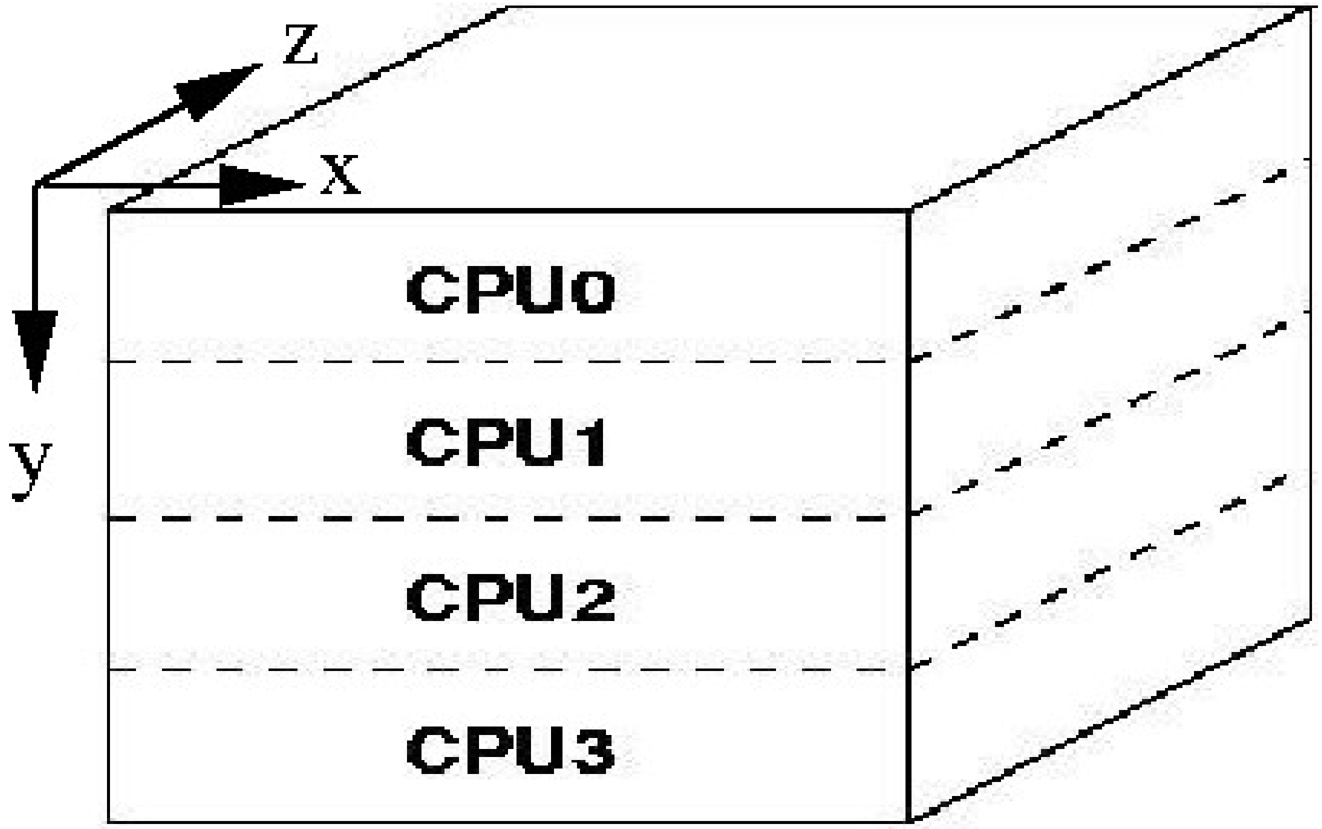
We need PARALLELIZATION !

Parallelization :

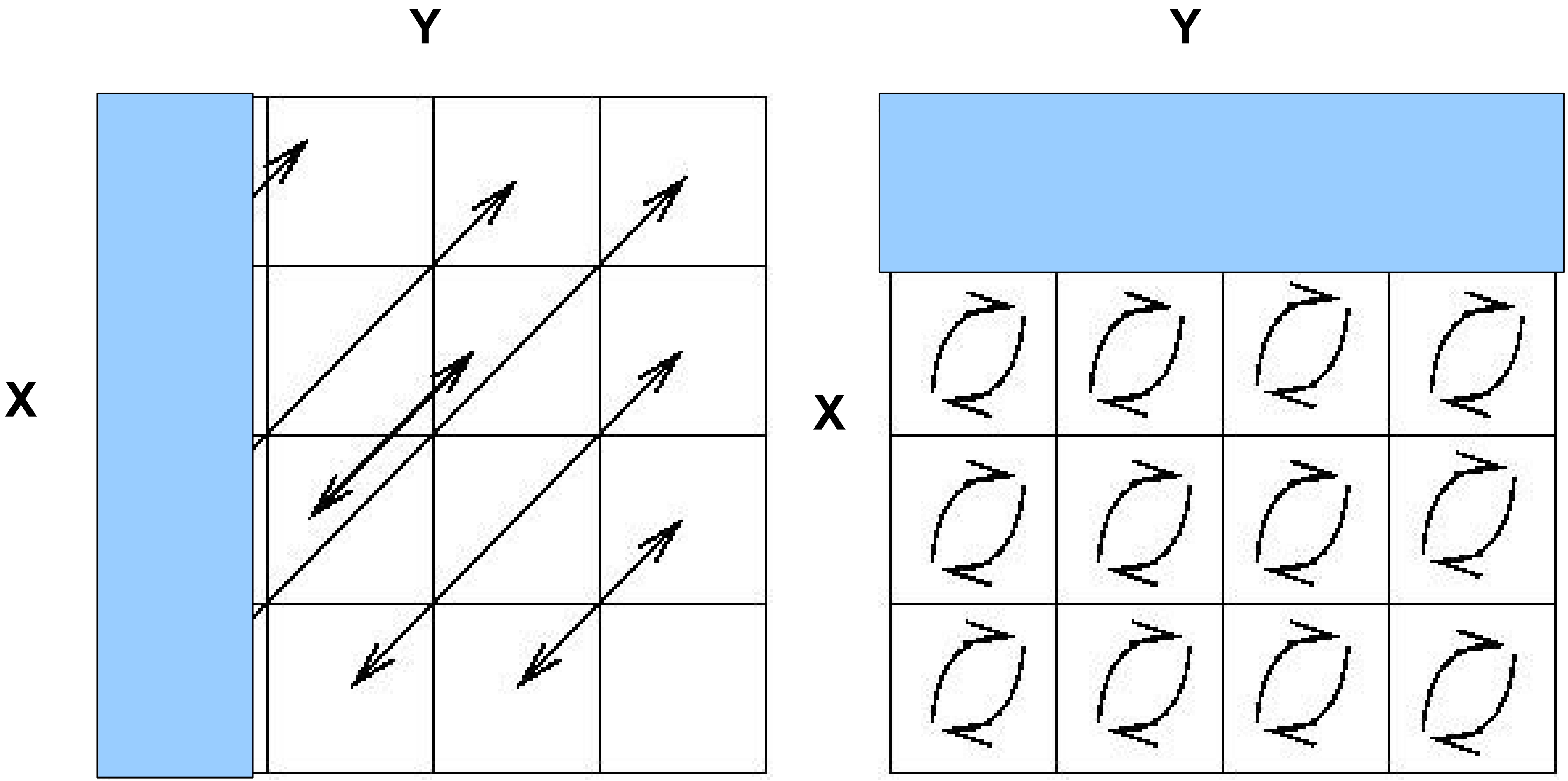
MPI : Message Passing Interface

MPI
MPI2

Z and X direction local



Transposition :



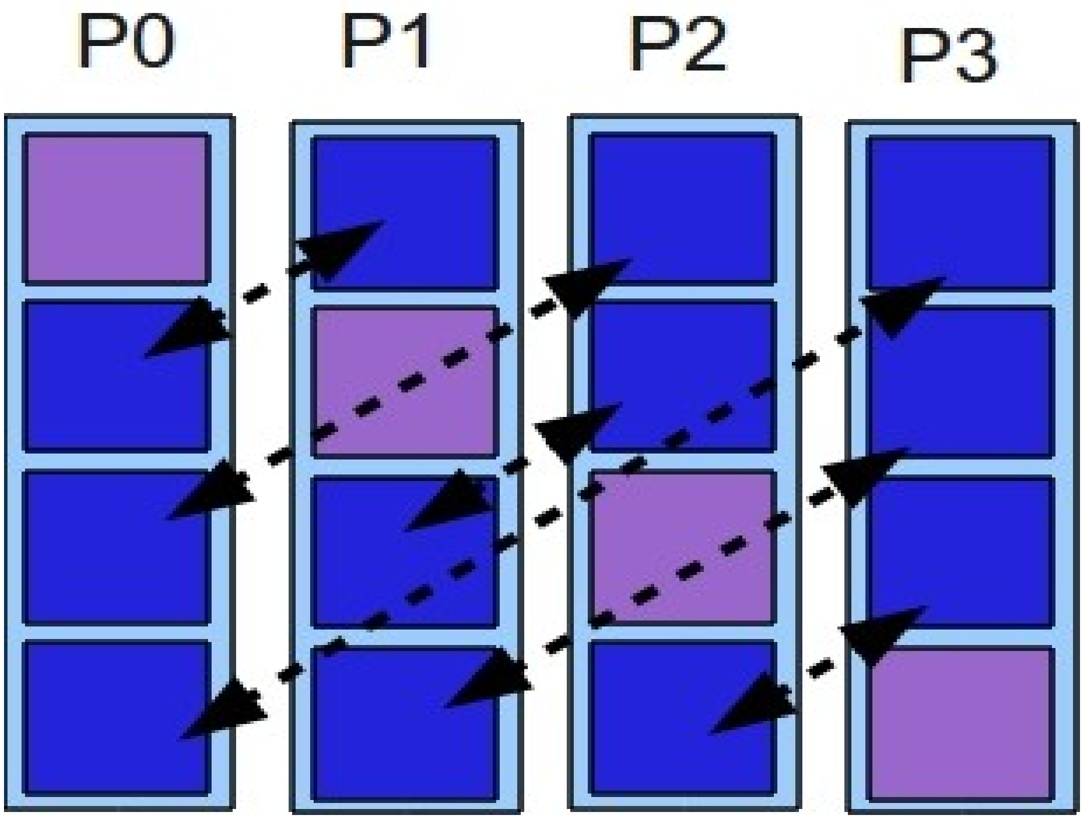
Global transpose of blocks

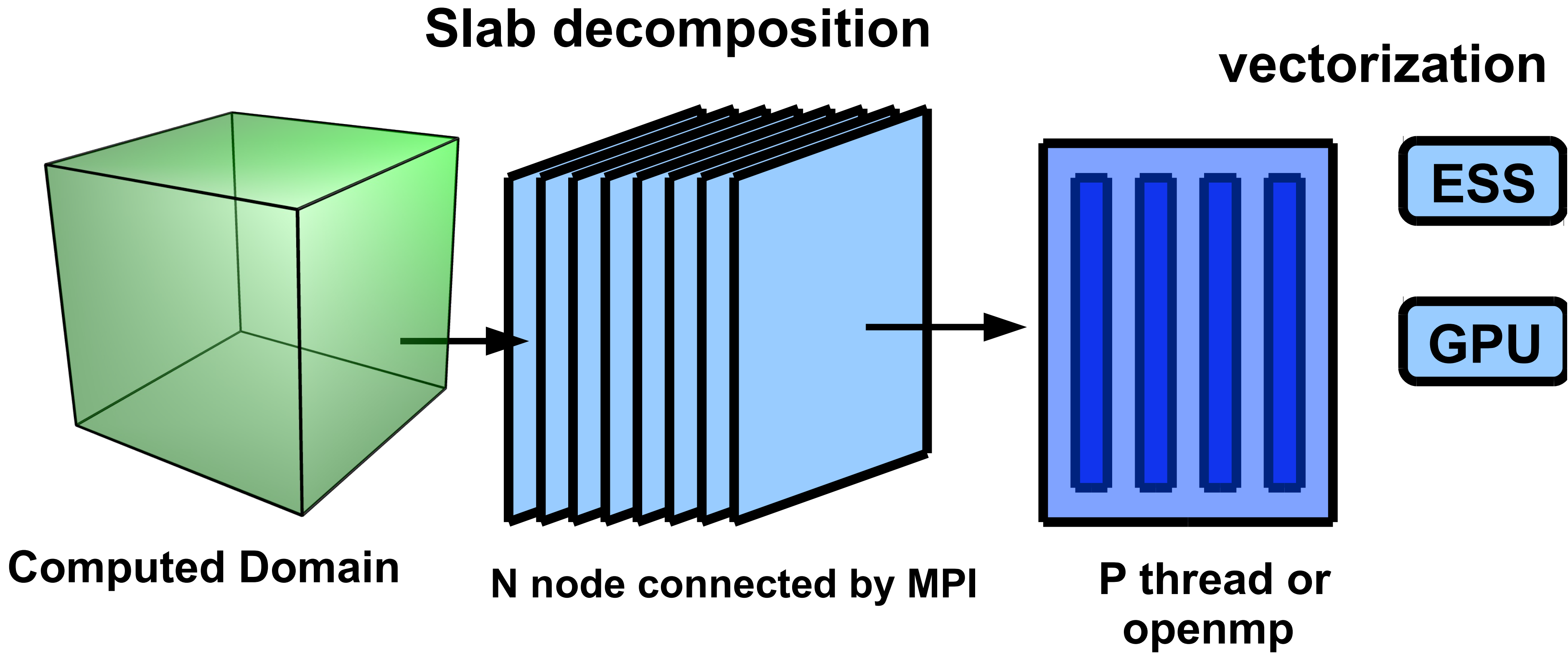
Local transposes within blocks

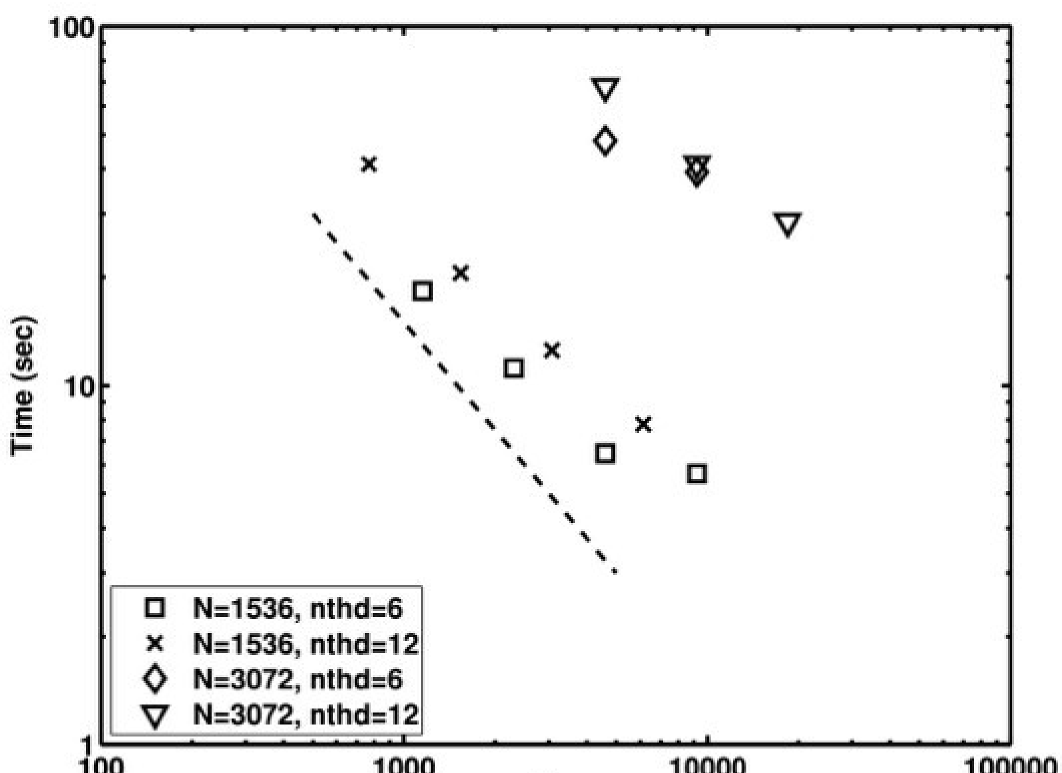
Recommend : www.fftw.org

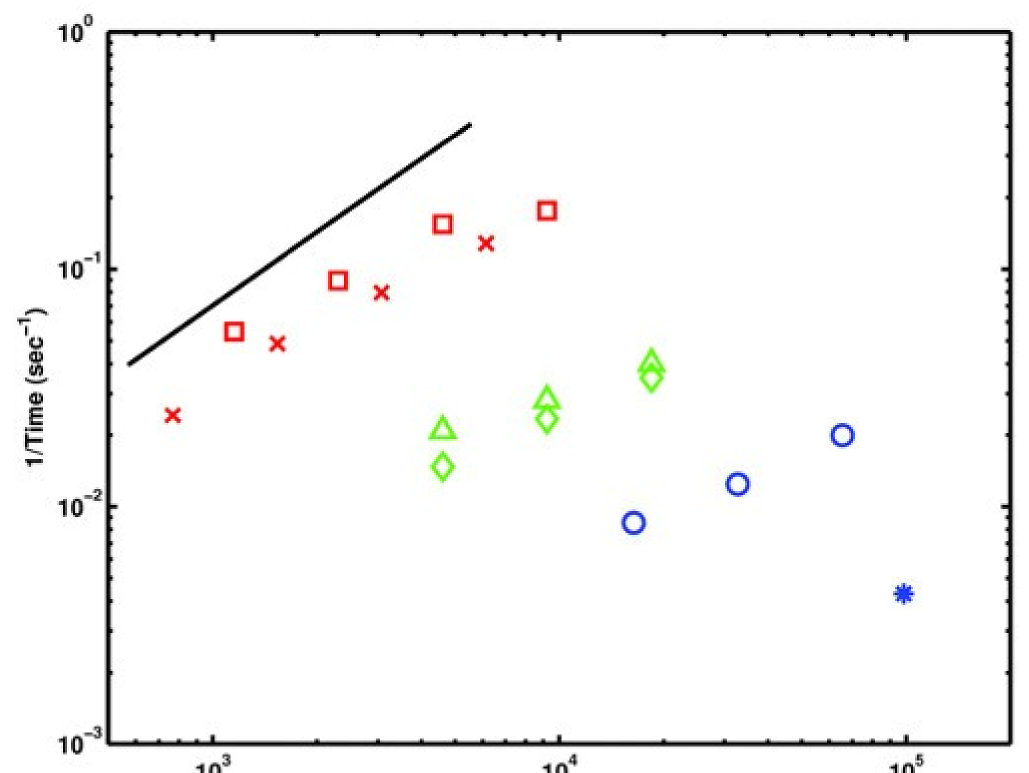
FFT 3D → Transposition of the cube !

maximum moving data algorithm

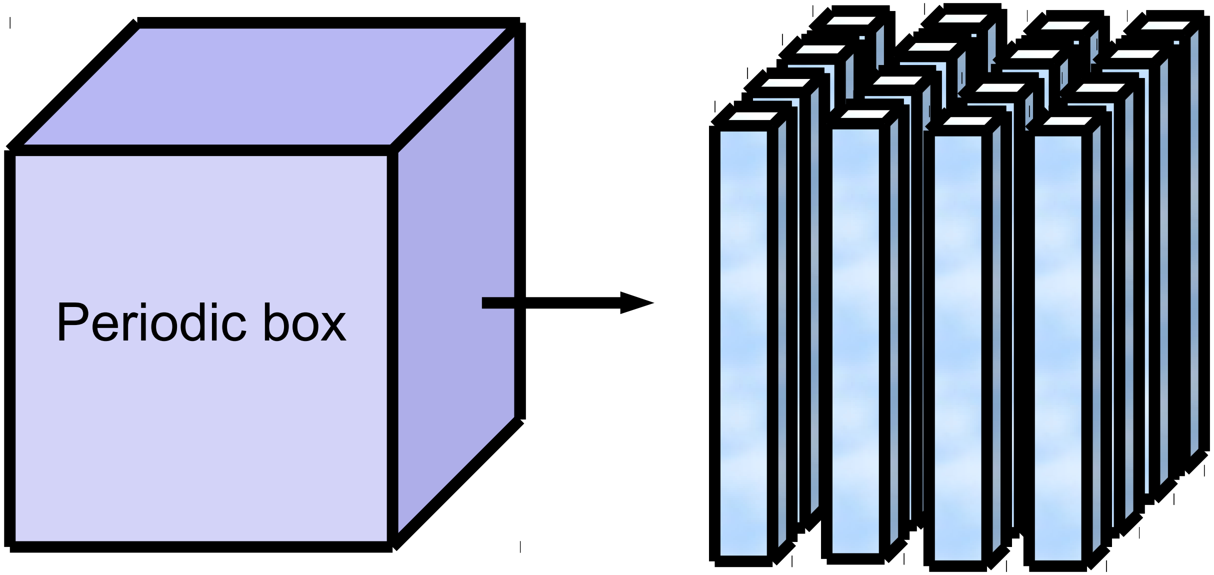



Slab decomposition
vectorization
ESS
GPU
Computed Domain
N node connected by MPI
P thread or
openmp


Time (sec)
100
10
1
100
1000
10000
100000
N=1536, nthd=6
N=1536, nthd=12
N=3072, nthd=6
N=3072, nthd=12

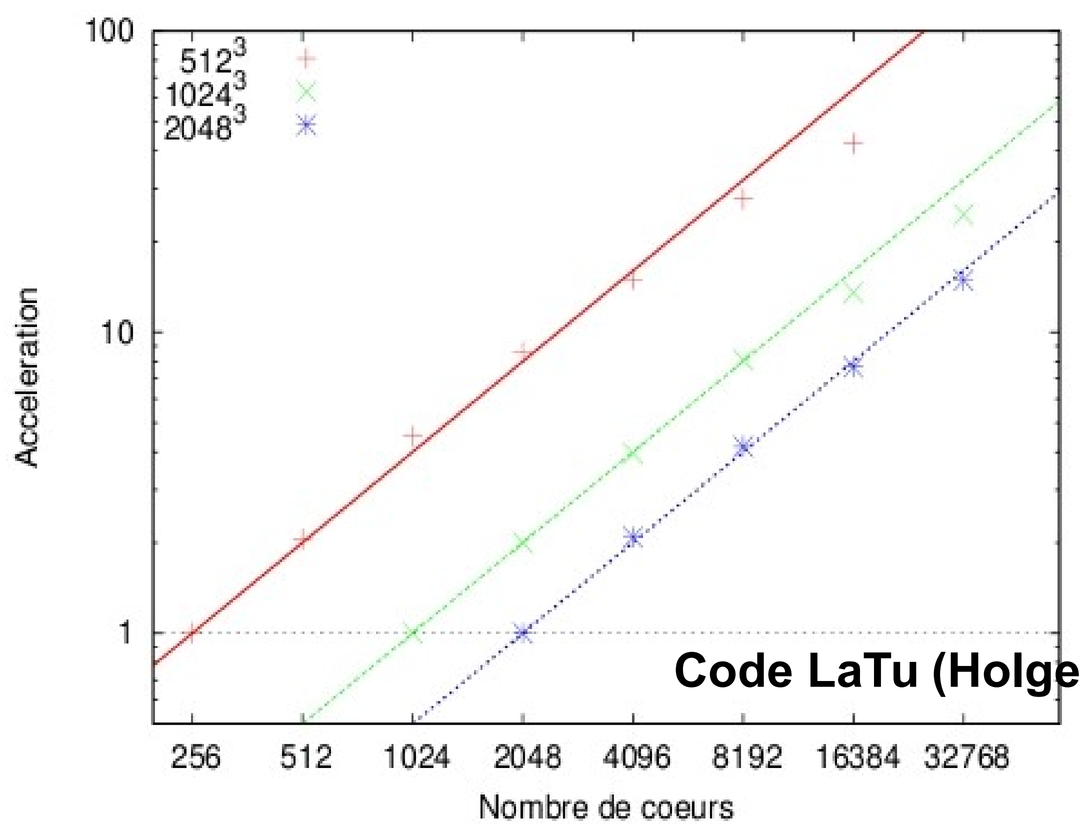

1/Time (sec⁻¹)

French Fries configuration



Computed Domain

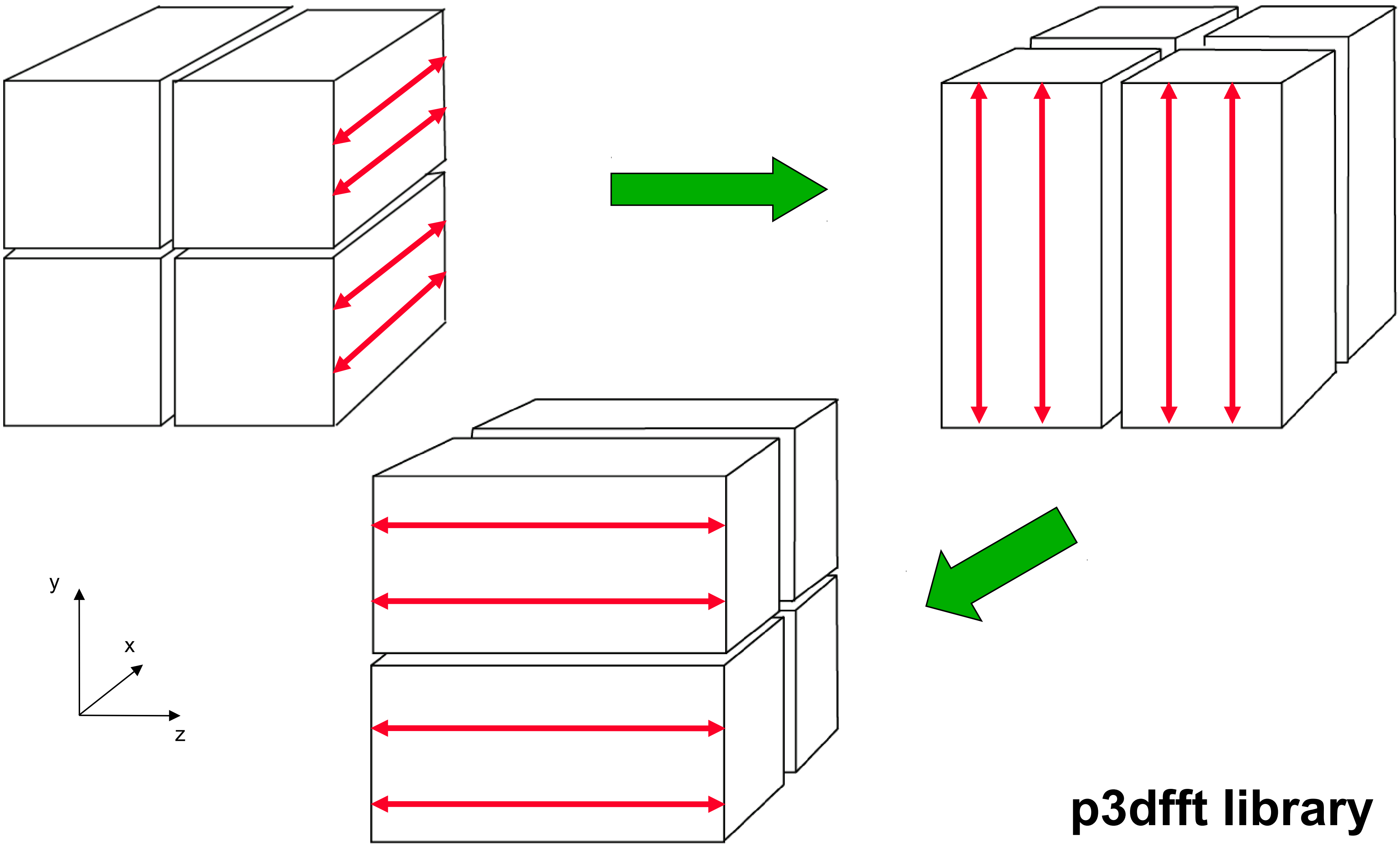
Pencil strategy



Code LaTu (Holger Homann)

http://www.sdsc.edu/us/resources/p3dfft.php

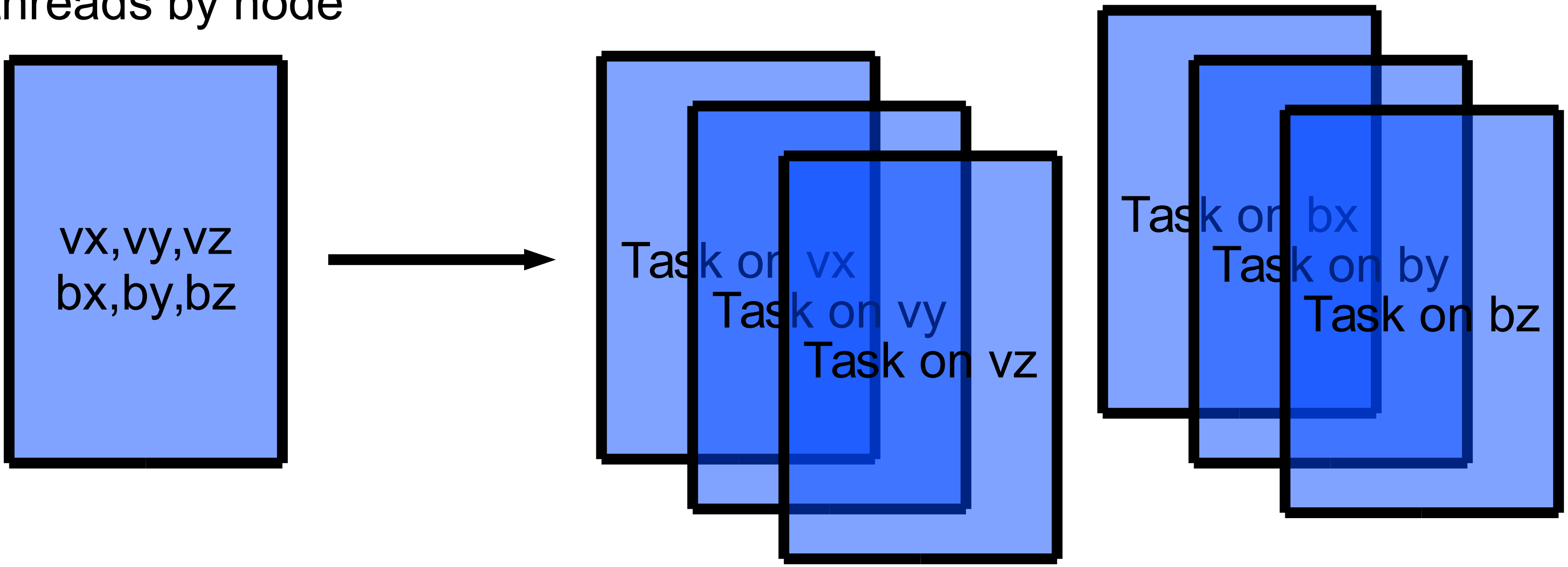
2D decomposition



p3dfft library

MPI and Multi-Threading task strategies

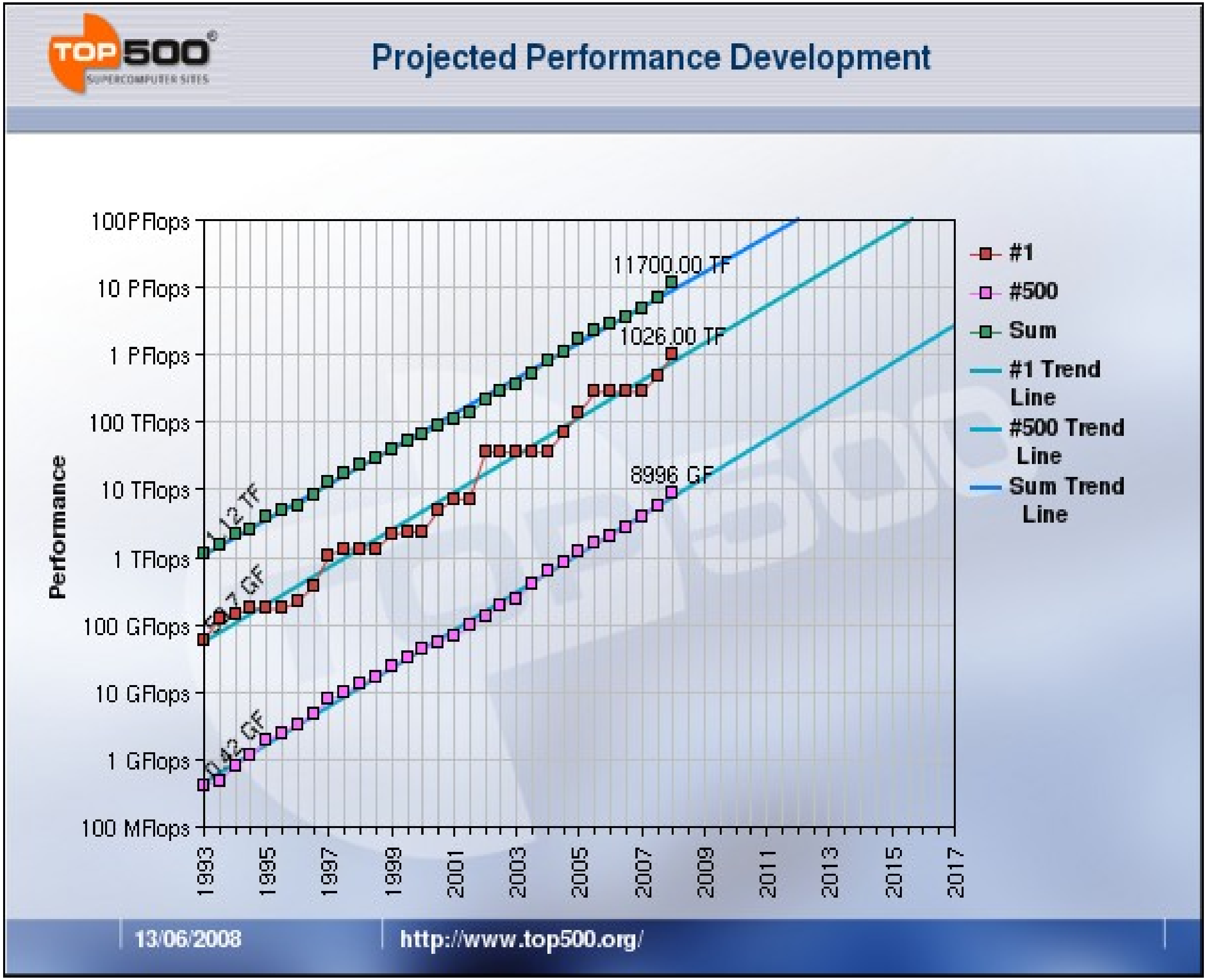
6 threads by node



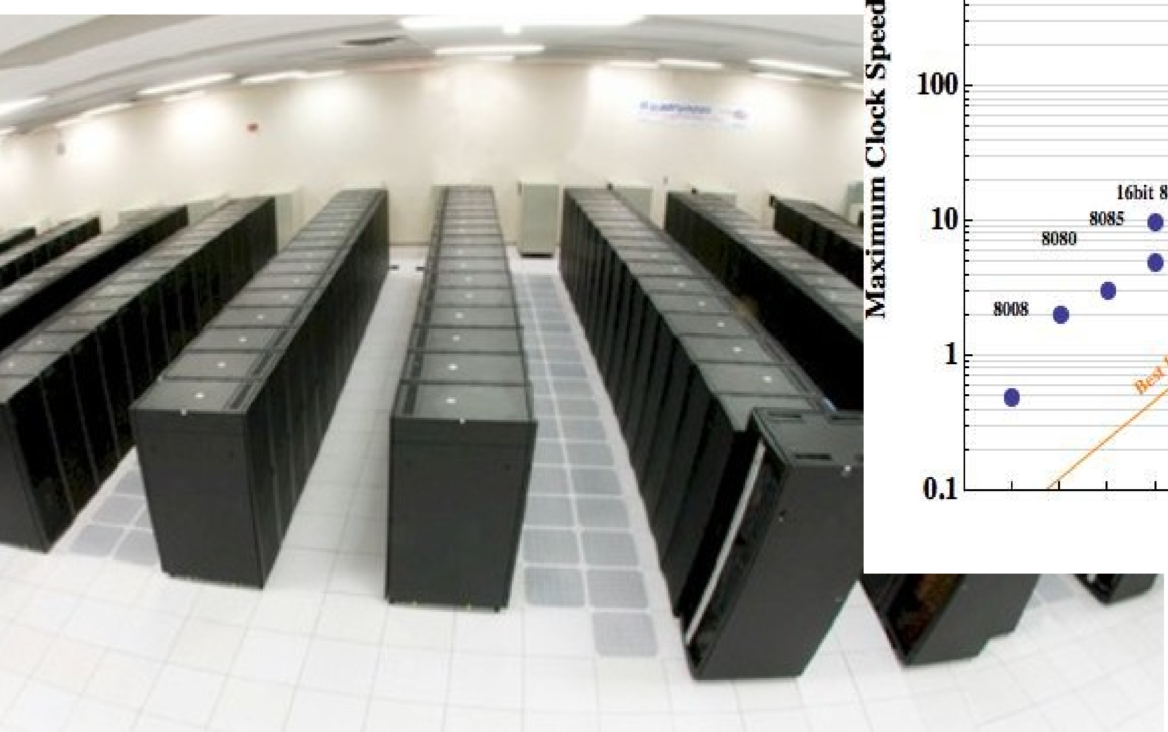
Strategy 2 → cut in 6 task, 1 task by thread

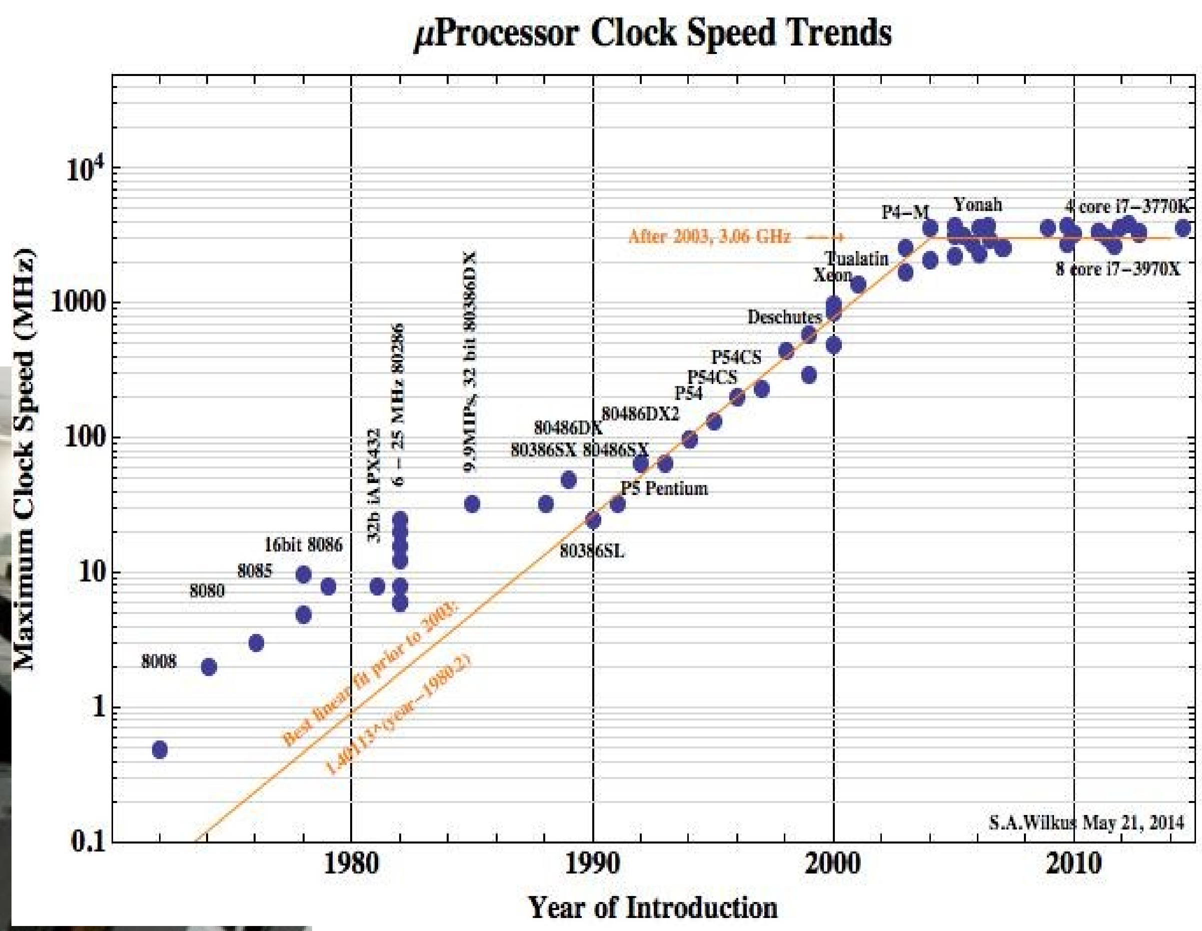
Overlap : Computation and communication in the transposition !

www.top500.org

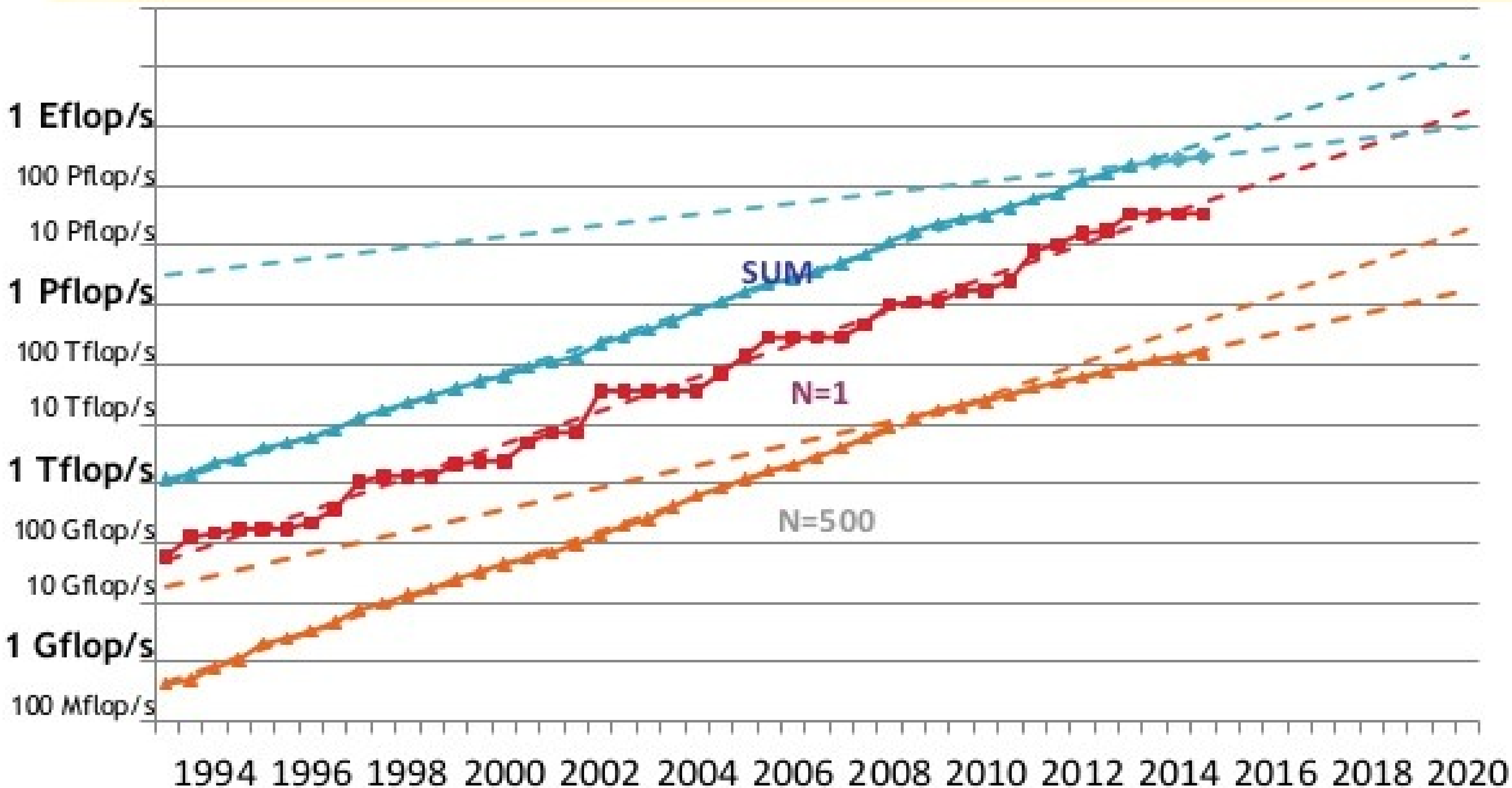


Why many core ?





PROJECTED PERFORMANCE DEVELOPMENT



Code C++

Dependence Tree

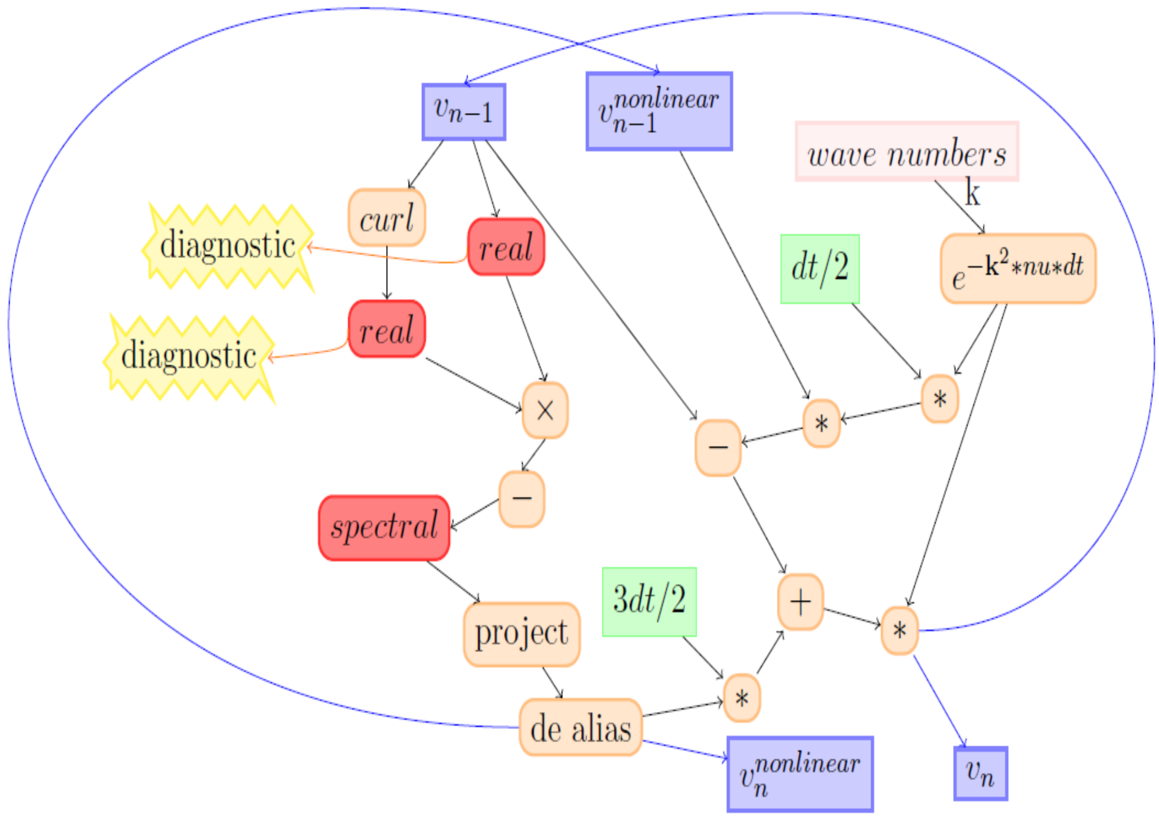


Figure 8: Basic dependence tree for a simplified simulation loop.

High level library « for physicist »

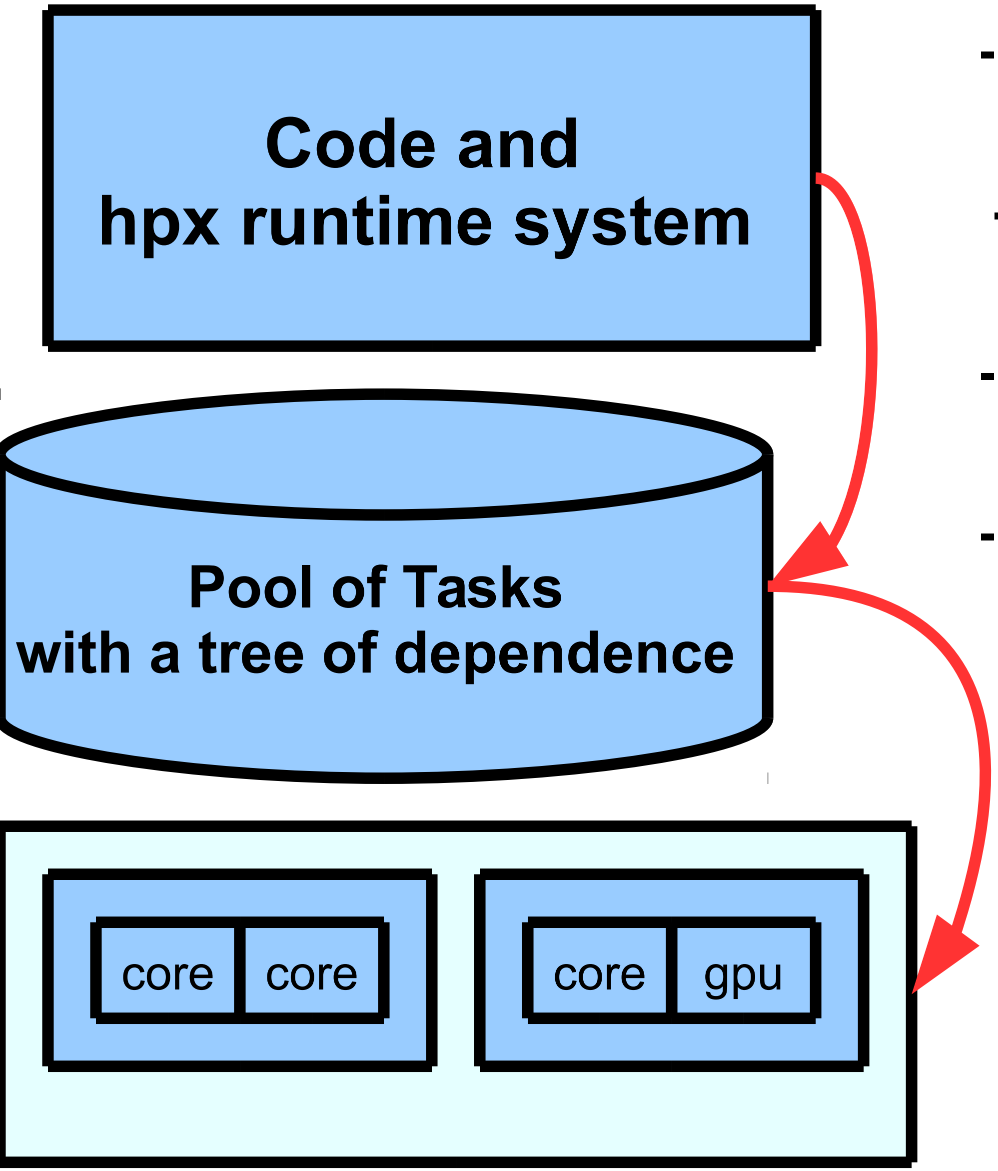
Low level library Parallelisation MPI, thread, GPU

Tasks management

Overlap : computation and communication !

(High Performance ParalleX)

c++ runtime system for parallel and distributed applications of any scale.
It strives to provide a unified programming model which transparently utilizes the available resources to achieve unprecedented levels of scalability.



- **Embrace Fine-grained Parallelism instead of Heavyweight Threads**
- **Rediscover Constrained Based Synchronization to replace Global Barriers**
- **Prefer Moving Work to the Data over Moving Data to the Work**
- **Favor Message Driven Computation over Message Passing**

http://stellar.cct.lsu.edu/projects/hpx/

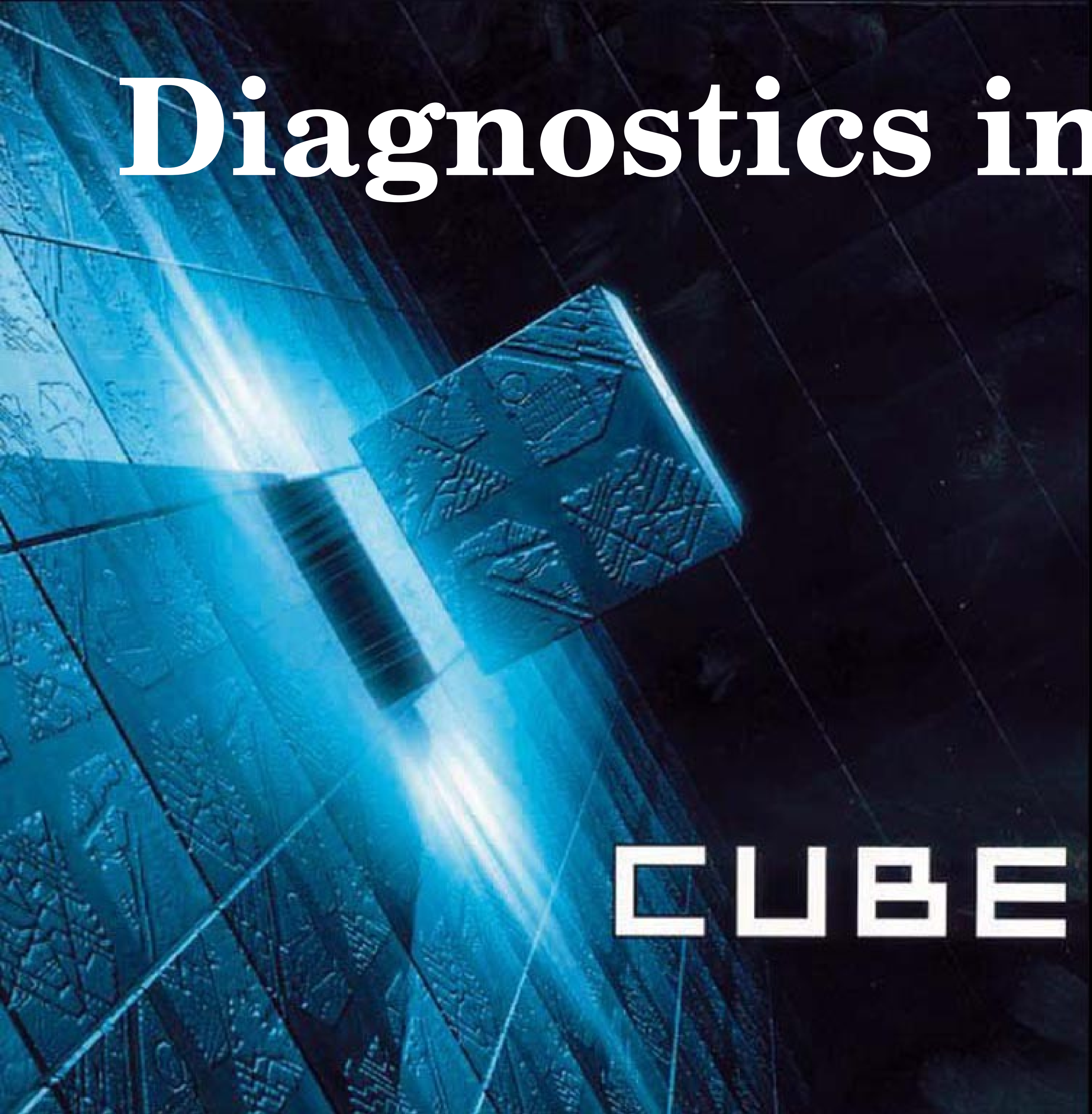
Perspectives in parallelization

Overlapping computation, communication, advanced diagnostics, IO, multiphysics

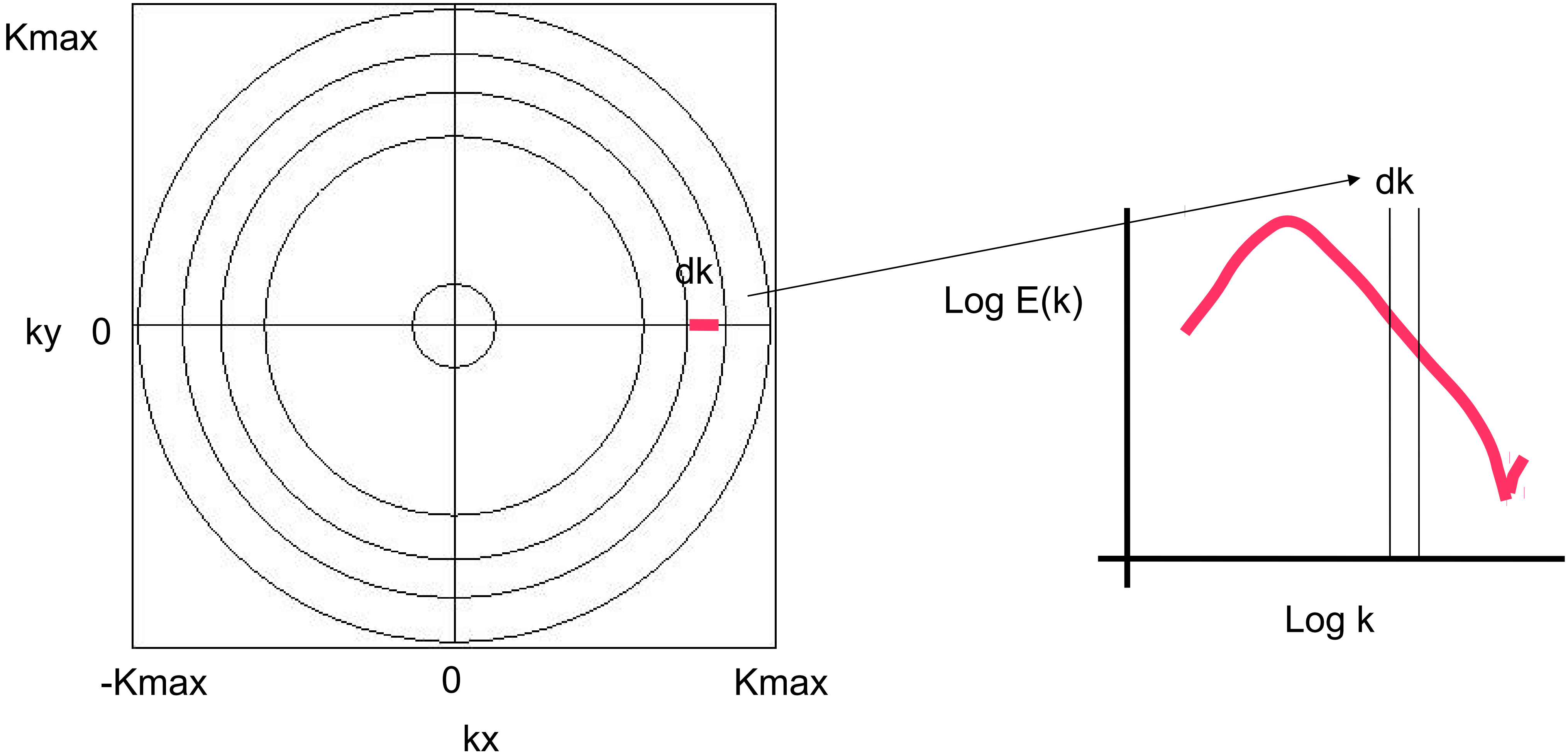
Challenge : Resolution of course
But also long run for thousand of eddy turn of time !

Do Spectral methods are really for Exascale computer ?

Diagnostics in the



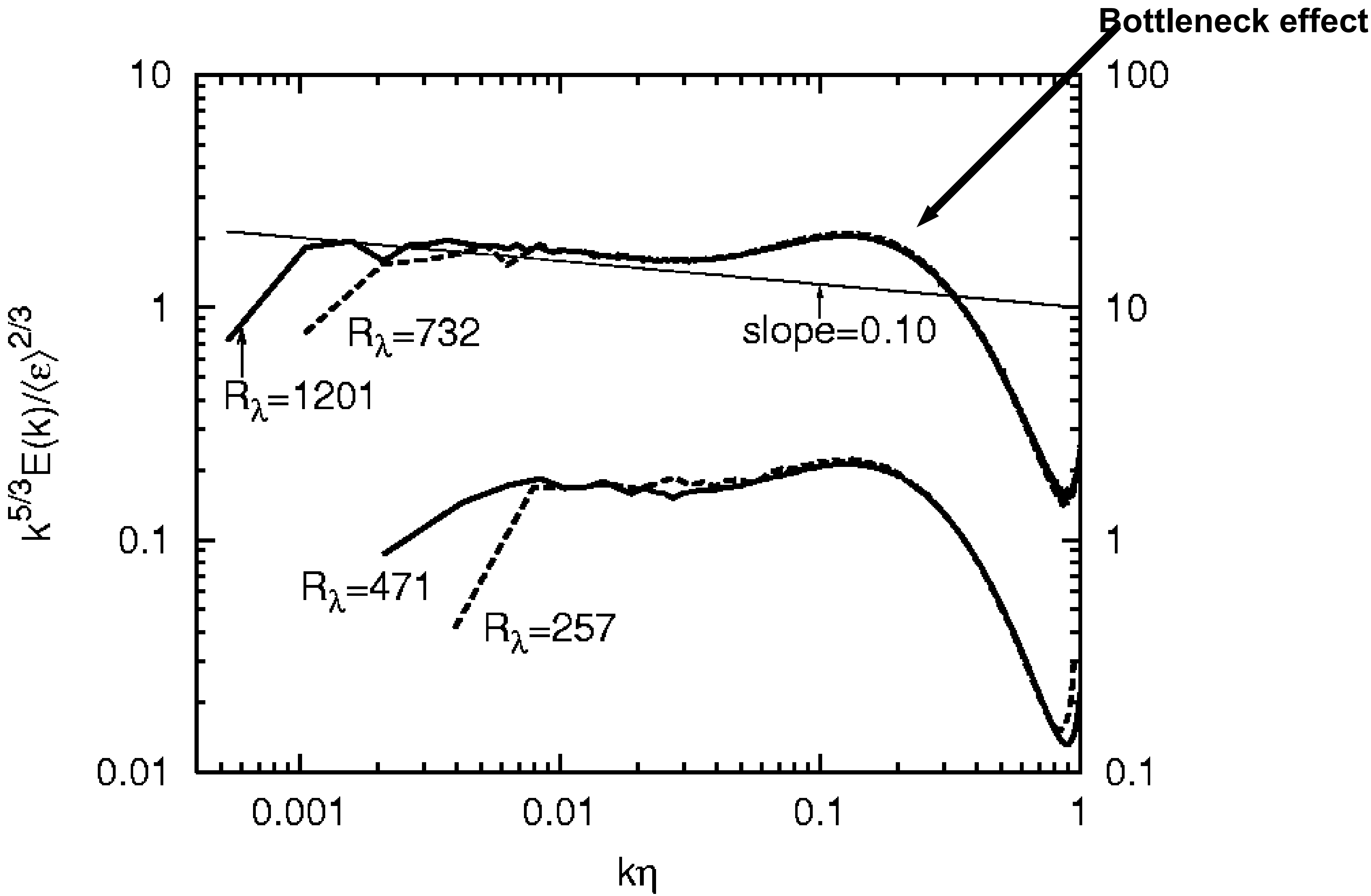
1 D spectra from 3D velocity field :



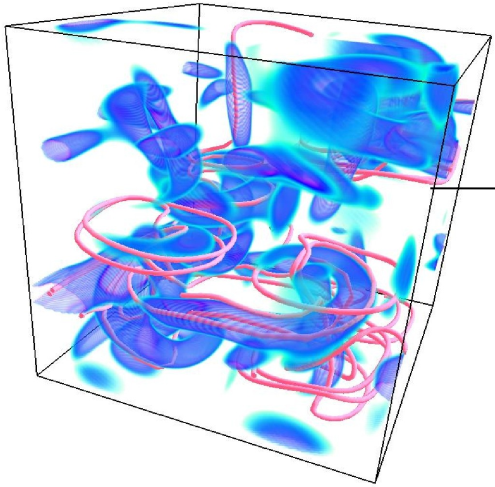
$$E = \sum E(k) = \sum \frac{1}{2}\left(\hat{u}_x(k)^2 + \hat{u}_y(k)^2 + \hat{u}_z(k)^2\right) dk$$

$$L_i = \frac{\sum E(k)/k}{\sum E(k)}$$

Y. Kaneda et al *PHYSICS OF FLUIDS* ***15*** *2003*



Numerical quantities output and experimental observable



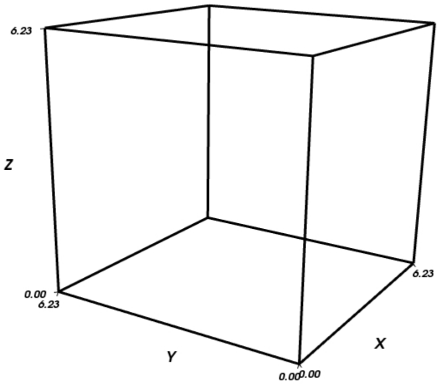
1 D spectra (u, b, w, j, ...)
Snapshot 3D fields -> 3D visualization
Probes -> time signal
Average in time fields

Numerical Experiment output :

$$\int \vec{u}_{Rv}(\vec{r},\tau)\,d\tau \rightarrow \langle \vec{U}(\vec{r}) \rangle$$

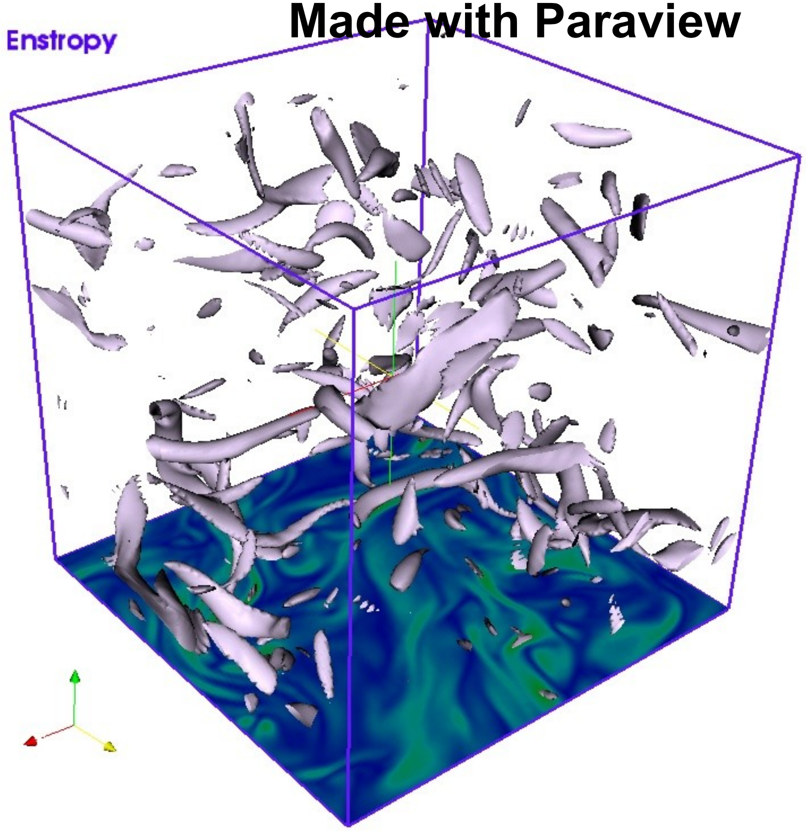
Average in time : $\sum \hat{u}_i(\vec{x},t_n) = \hat{u}_i(\vec{x})$

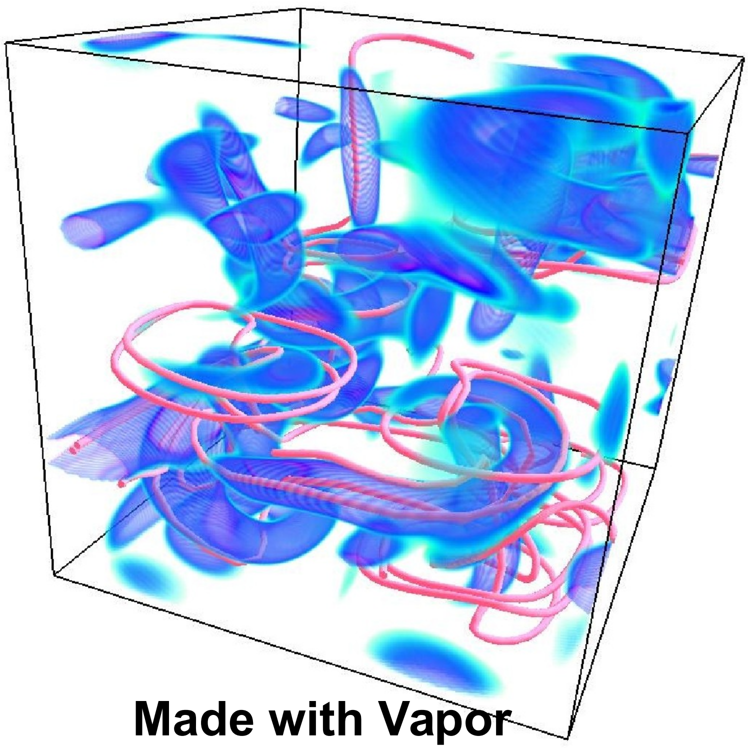
Probes : fix location , line -> signal



visualisation 3D

Made with Paraview





Made with Vapor

vapor.ucar.edu
www.paraview.org

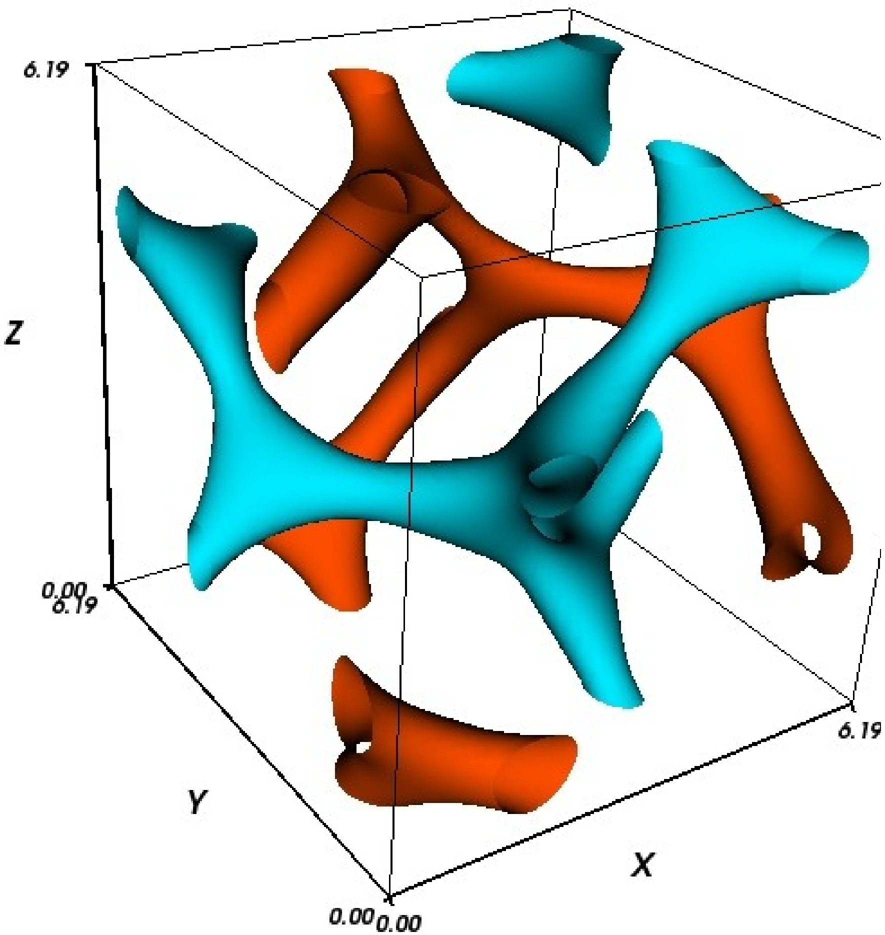
→

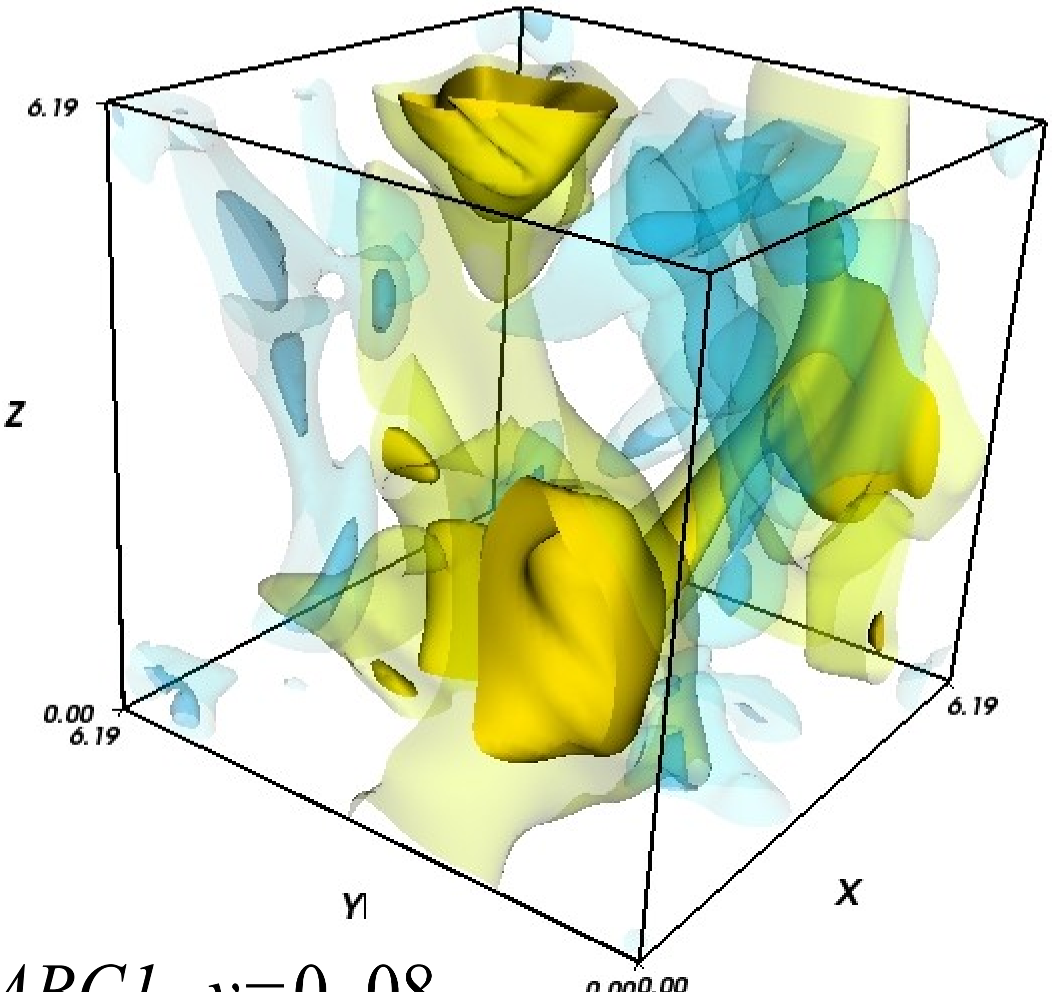
Large data set : wavelet decomposition
volume rendering, field line, periodic box
spherical geometry, connection with IDL

Average in time

$$\int \vec{u}_{Rv}\left(\vec{r},\tau\right)d\tau \rightarrow \left\langle \vec{U}\left(\vec{r}\right)\right\rangle$$

Average in time : $\sum \hat{u}_i\left(\vec{x},t_n\right)=\hat{u}_i\left(\vec{x}\right)$

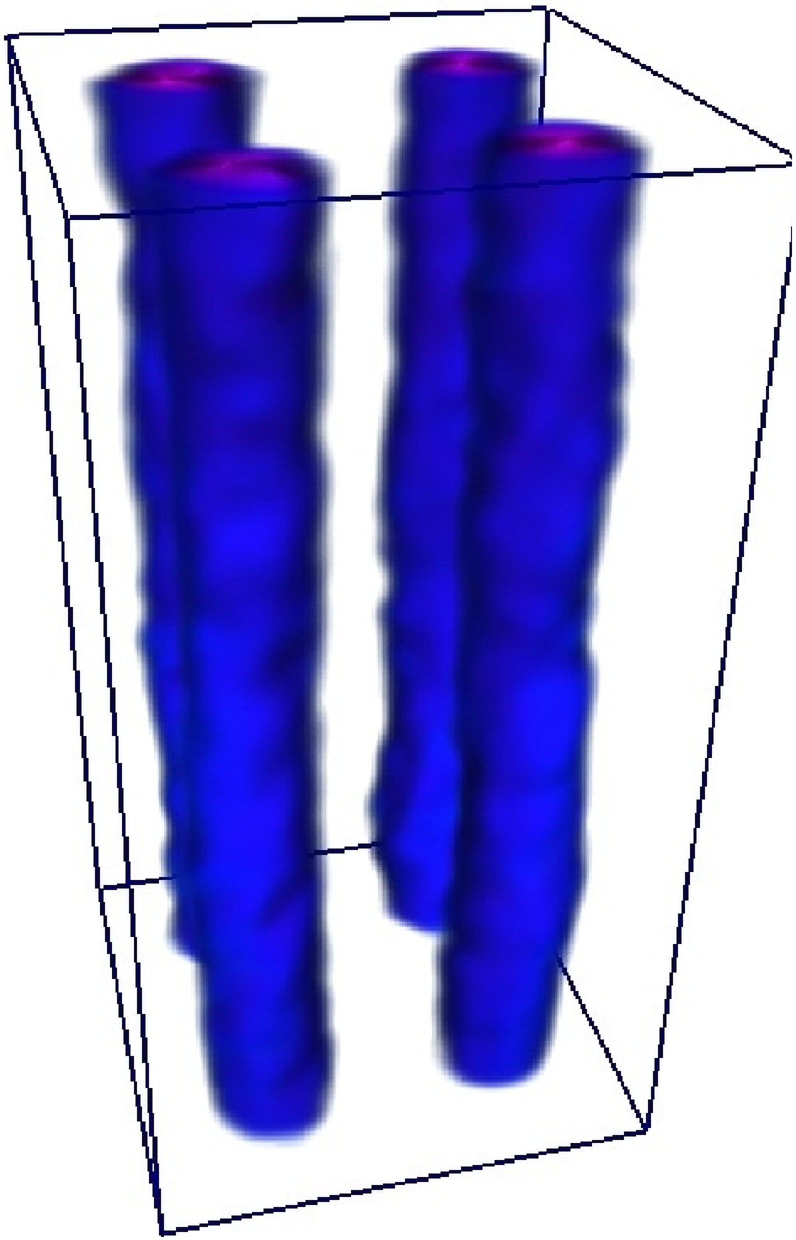




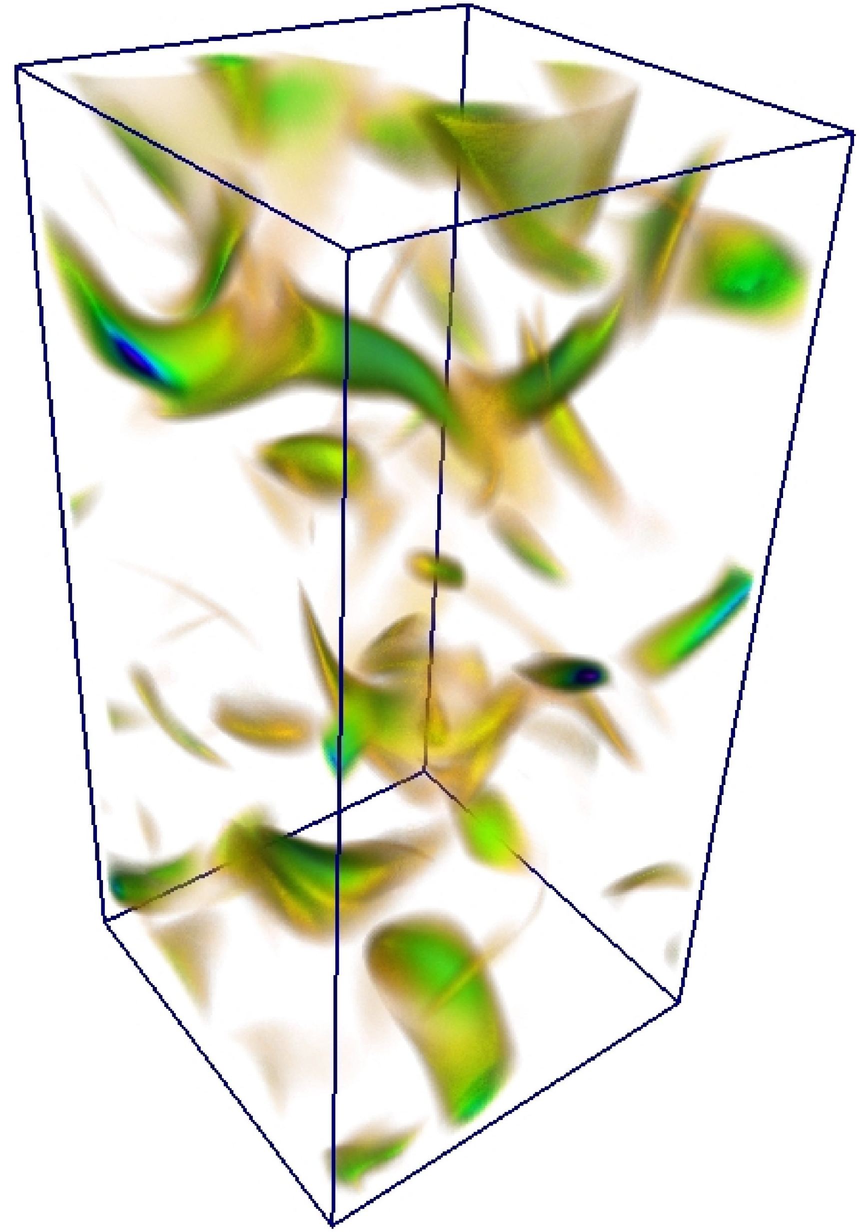
$$F=ABC1 \quad \nu=0.08$$

Made with paraview

G. O Robert Forcing



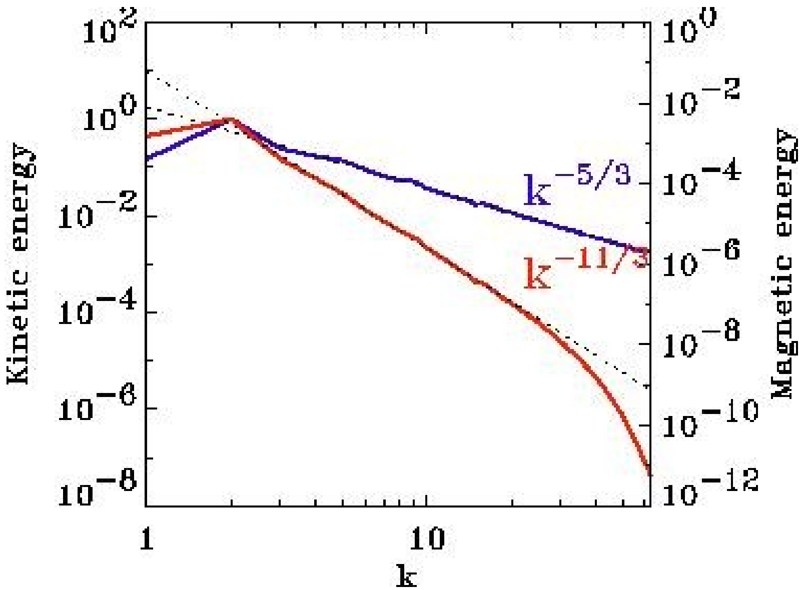
Average in time



Snapshot

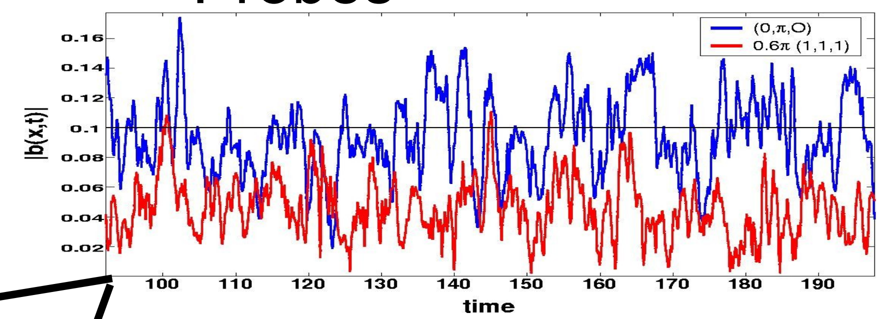
Made with Vapor

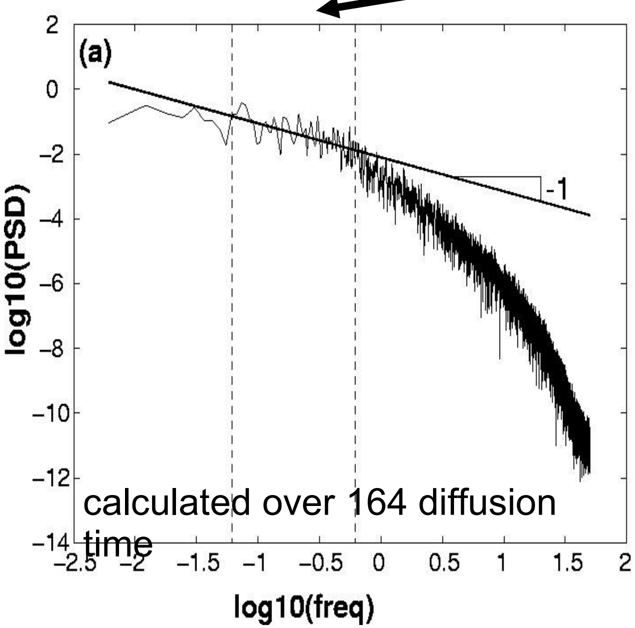
Magnetic induction results : Y.Ponty, H. Politano & J-F Pinton , *Phys. Rev. Lett.* 92 (2004)

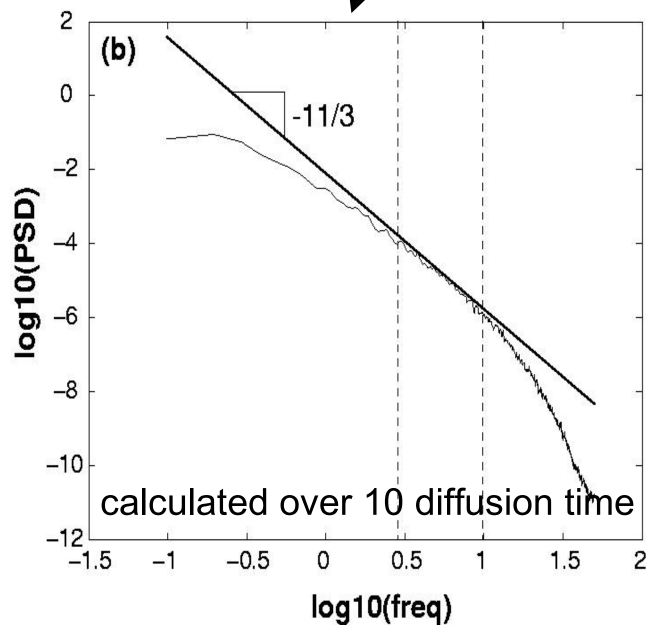


$$P_m \sim 10^{-3} - 10^{-4}$$

Probes







Power spectra of the temporal fluctuations
filtered by an Hanning window box

The 1/f power law has been found in the most of the experimental MHD measurements.

Y.Ponty

Euleriann & Lagrangian perspective

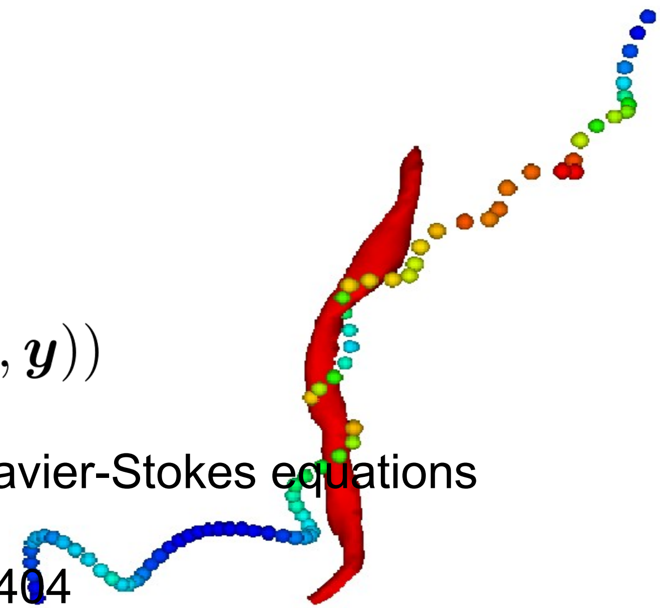
- How to define now the three different regimes?
- What do the fluid trajectories look like?
- Where does the magnetic field grow ?

Frame of reference

- Euler = Laboratory
- Lagrange = Particle

$$\frac{d\boldsymbol{X}(t,\boldsymbol{y})}{dt} = \boldsymbol{v}(\boldsymbol{X}(t,\boldsymbol{y}))$$

$\boldsymbol{v}(\boldsymbol{X}(t,\boldsymbol{y}))$ by Navier-Stokes equations



Fluid Turbulence

Toschi F and Bodenschatz E **2009** Ann. Rev. Fluid Mech. 41 375–404
Biferale Let al 2004 Phys. Rev. Lett. 93 4502
Yeung P K and Sawford M S **2006** J. Turbulence 7 1–12
Bec J et al **2006** J. Fluid Mech. 550 349–58
Yeung P K and Borgas M S **2004** J. Fluid Mech. 503 93–124
....

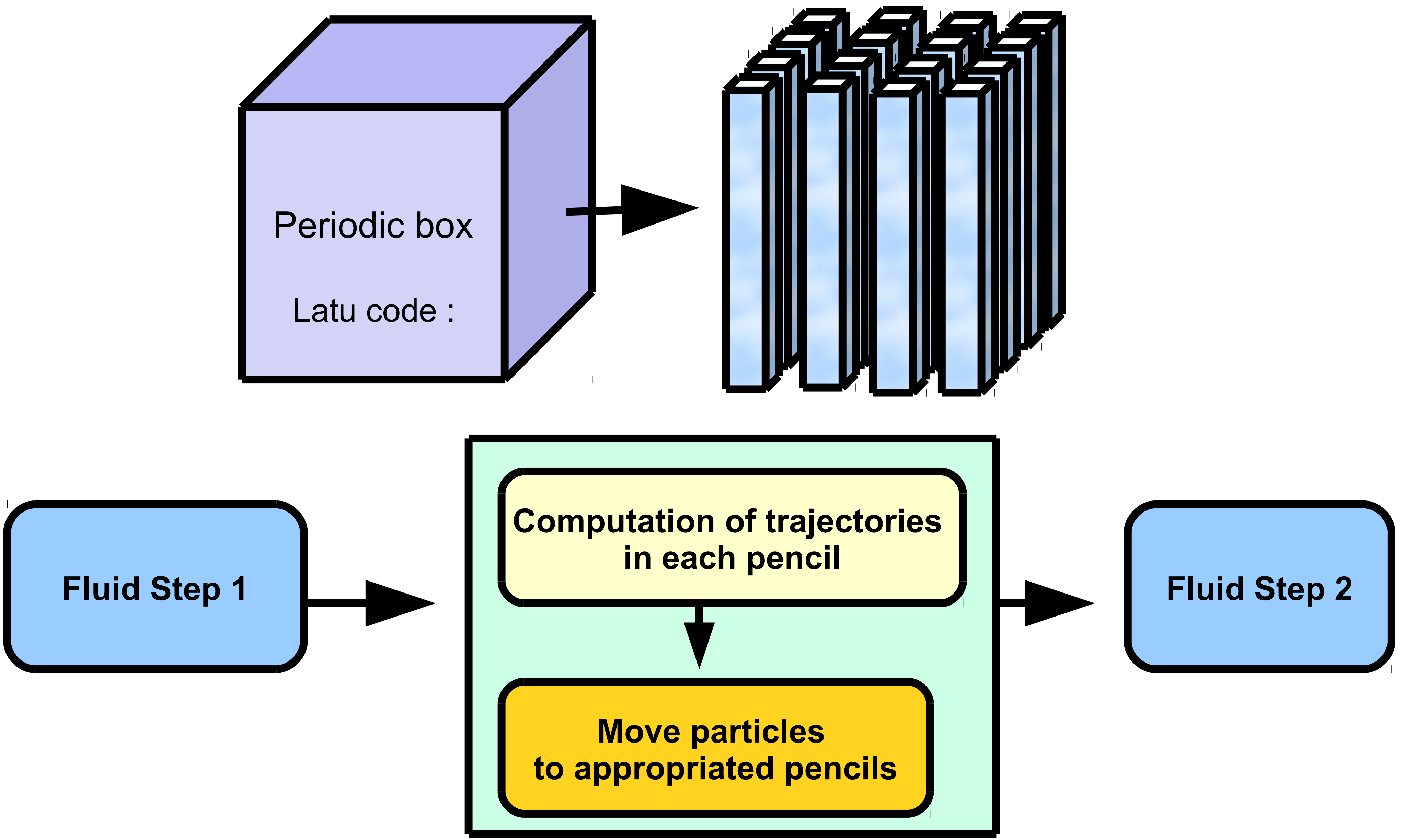
MHD Turbulence

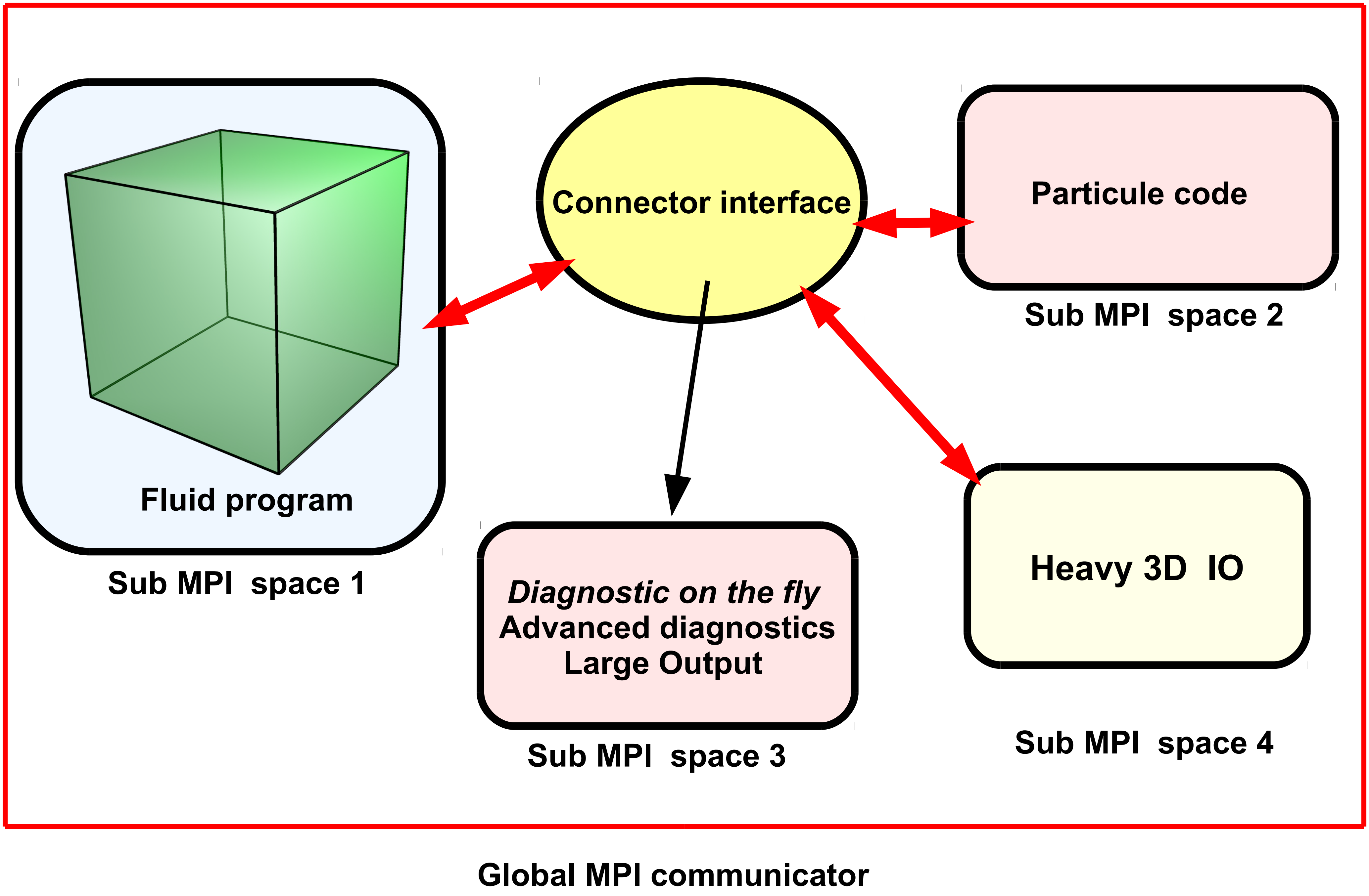
Homann H et al R **2009** Phys. Plasma 16 082308
Homann H et al **2007** J. Plasma Phys. 73 821–30
Homann H et al **2009** New J. Phys. 11 073020

Dynamo :

Holger Homann, Yannick Ponty, Giorgio Krstulovic, Rainer Grauer
"Structures and Lagrangian statistics of the Taylor-Green Dynamo"
New J. Phys. **16** 075014 **(2014)**
doi:10.1088/1367-2630/16/7/075014

Parallel strategy for the tracers & particles




Connector interface
Particule code
Sub MPI space 2
Fluid program
Sub MPI space 1
Diagnostic on the fly
Advanced diagnostics
Large Output
Sub MPI space 3
Heavy 3D IO
Sub MPI space 4
Global MPI communicator

Are we able to escape from the

Classic Penalization method

$$\partial_t \vec{u} + \vec{u}\nabla\vec{u} = -\nabla P + \nu\Delta\vec{u} + \vec{j}\times\vec{b} - C\xi(\vec{x})\vec{u}$$

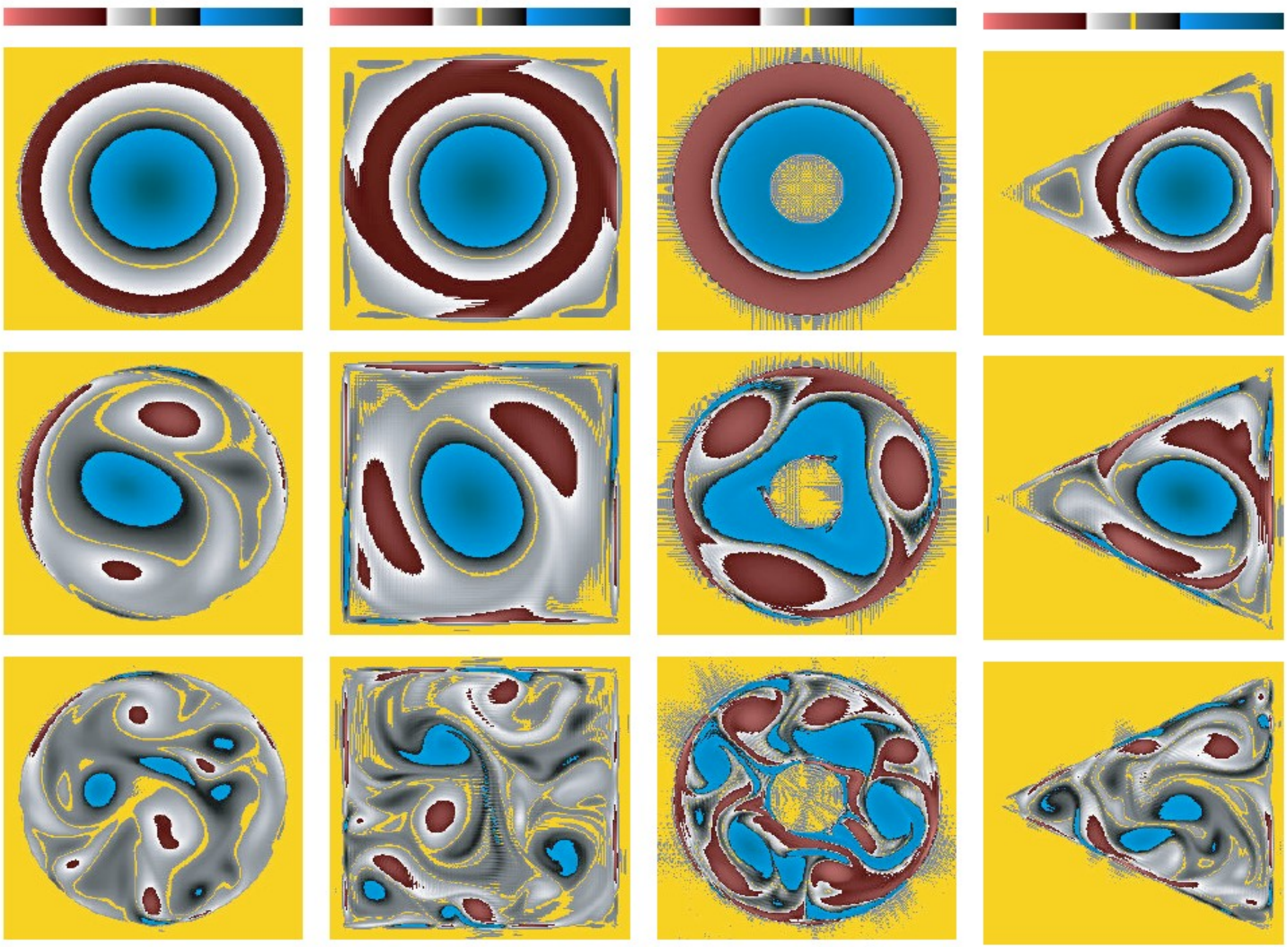
$$\nabla\vec{u} = 0$$

$$\xi(\vec{x}) = 1 \; \textit{inside boundary}$$

$$\xi(\vec{x}) = 0 \; \textit{outside boundary}$$

Kai Schneider[1] and Marie Farge[2]

Computational Physics and New Perspectives in Turbulence
Y. Kaneda (Ed.)
Springer, 2007, pp. 241-246



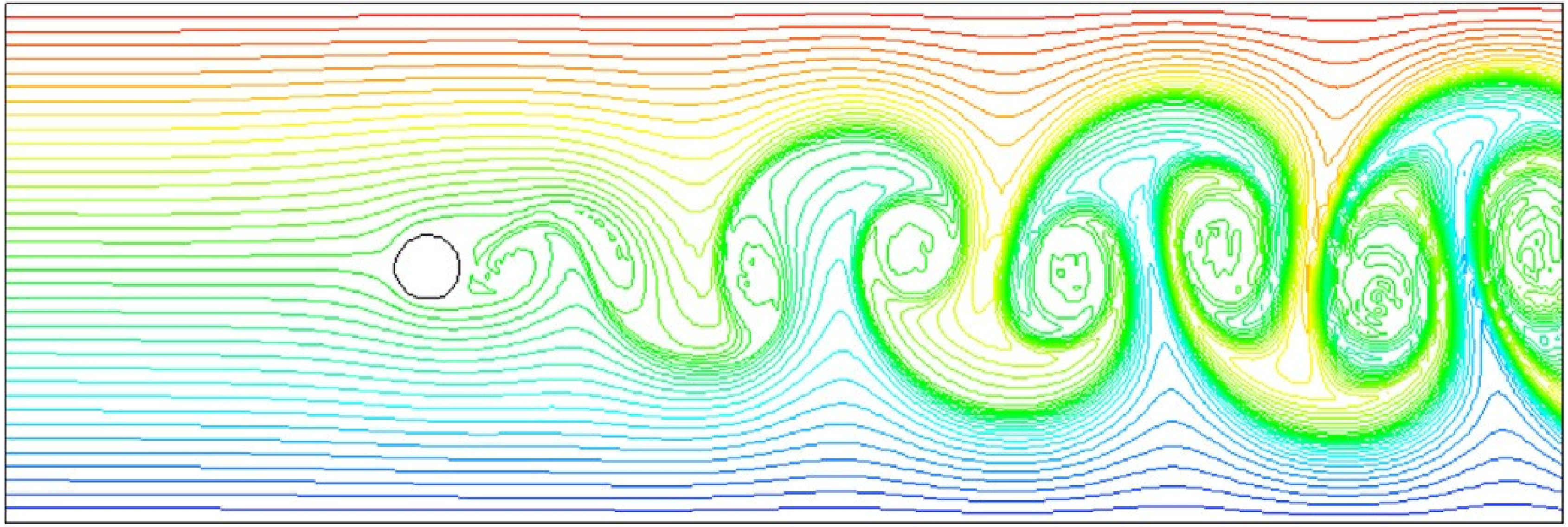
Pseudo-Penalization method

M. Minguez, R. Pasquetti and E. Serre
"A pseudo-penalization method for high Reynolds unsteady flows"
Applied Numerical Mathematics
Volume **58**, Issue 7, **July 2008**, Pages 946-954

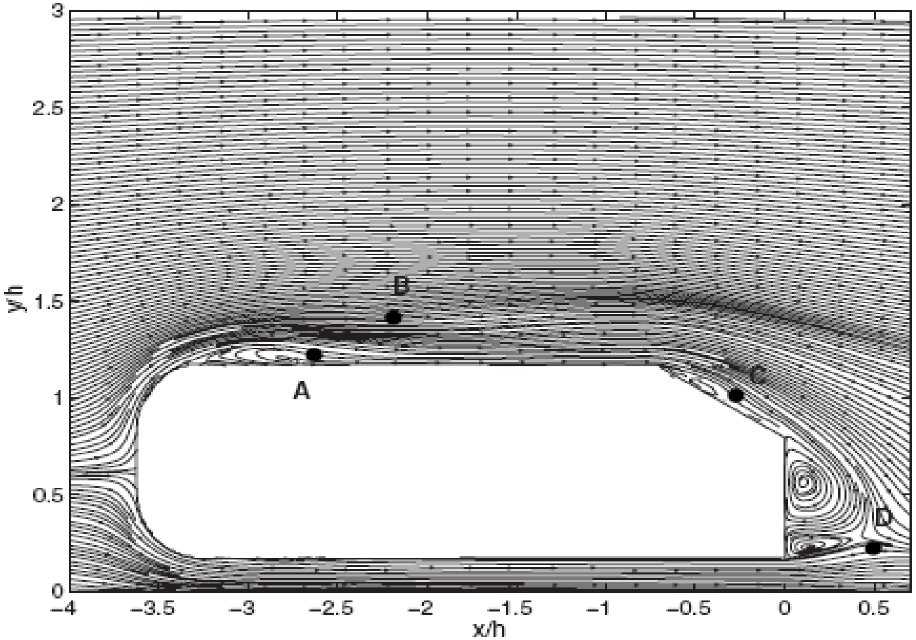
$$\nu \Delta u^{n+1} - \frac{\alpha}{\tau} u^{n+1} - \nabla p^{n+1} = (1-\chi) f^{n+1} \quad \text{in } \Omega,$$

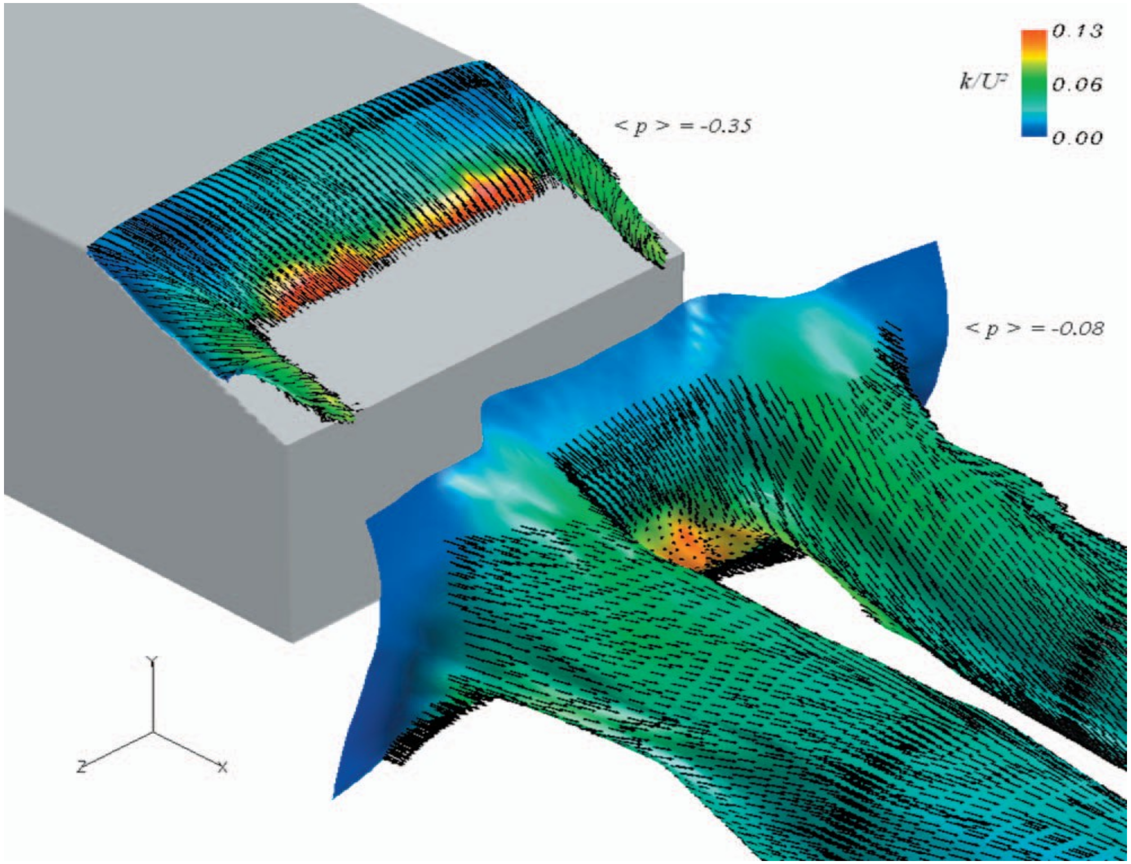
$$\nabla \cdot u^{n+1} = 0,$$

$$B\left(u^{n+1}\right) = g^{n+1} \quad \text{on } \Gamma,$$



M. Minguez, R. Pasquetti, and E. Serre
PHYSICS OF FLUIDS **20**, 095101 **2008**





Penalisation method : Direct forcing

E. A Fadlum et al JCP 161, 35–60 (2000)

$$\mathbf{u}^{(n+1)} = \mathbf{u}^{(n)} + \Delta t \cdot \left(\mathcal{L}\left(\mathbf{u}^{(n)}\right) + \mathbf{f}_b^{(n)}\right)$$

$$\mathbf{f}_b^{(n)} = -\chi_p(\mathbf{r},t)\{\mathcal{L}(\mathbf{u}^{(n)}) + \frac{1}{\Delta t}(\mathbf{u}^{(n)} - \mathbf{V}^{(n+1)})\}$$

J. Mohd-Yusof,
CTR Annual Research Briefs,
NASA Ames/Stanford University, (1997).

Volume Fraction method :

$$\overline{\chi}_p(\mathbf{r}) = \frac{1}{2^3 \Delta x \Delta y \Delta z} \int_{-\Delta x}^{\Delta x} \int_{-\Delta y}^{\Delta y} \int_{-\Delta z}^{\Delta z} \chi_p(\mathbf{r}+\mathbf{r}^*) \mathrm{d}^3 r^*$$

E. A Fadlum et al JCP 161, 35–60 (2000)

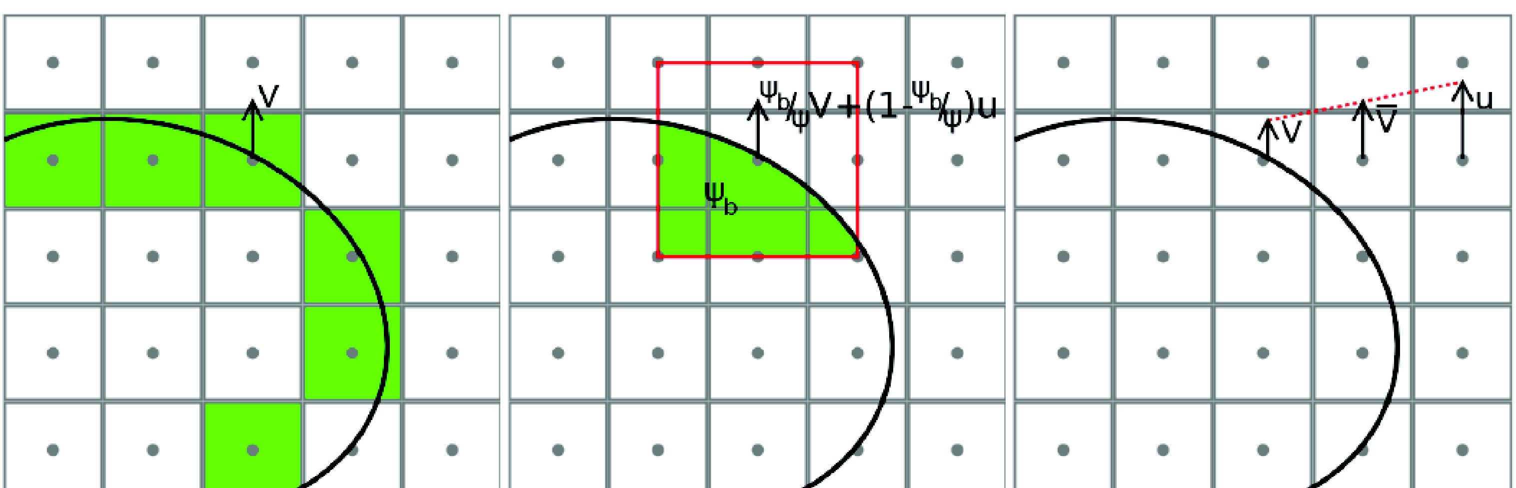


Fig. 3.2.: Stepwise interpolation, volume fraction method and linear interpolation

The pressure predictor

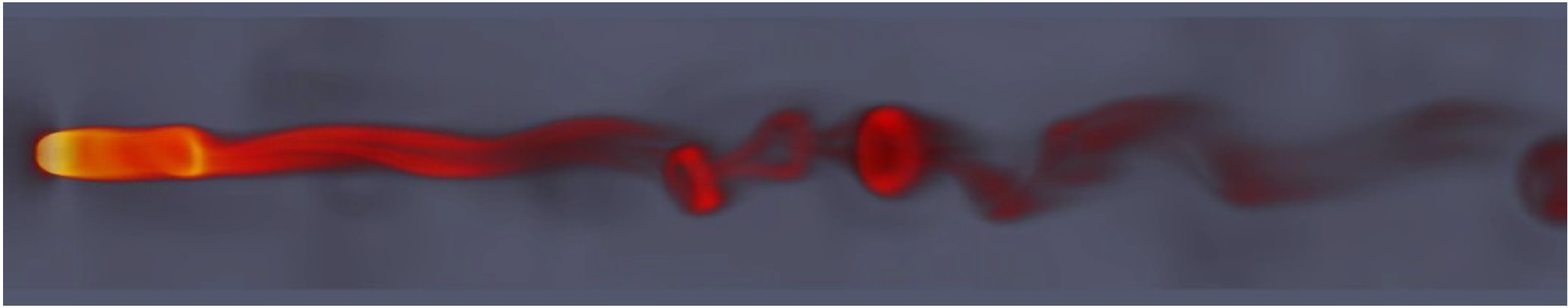
Brown, D. L., Cortez, R. & Minion, M. L. Accurate Projection Methods for the Incompressible Navier Stokes Equations. J. Comp. Phys. 168 (2), 464–499 (2001).

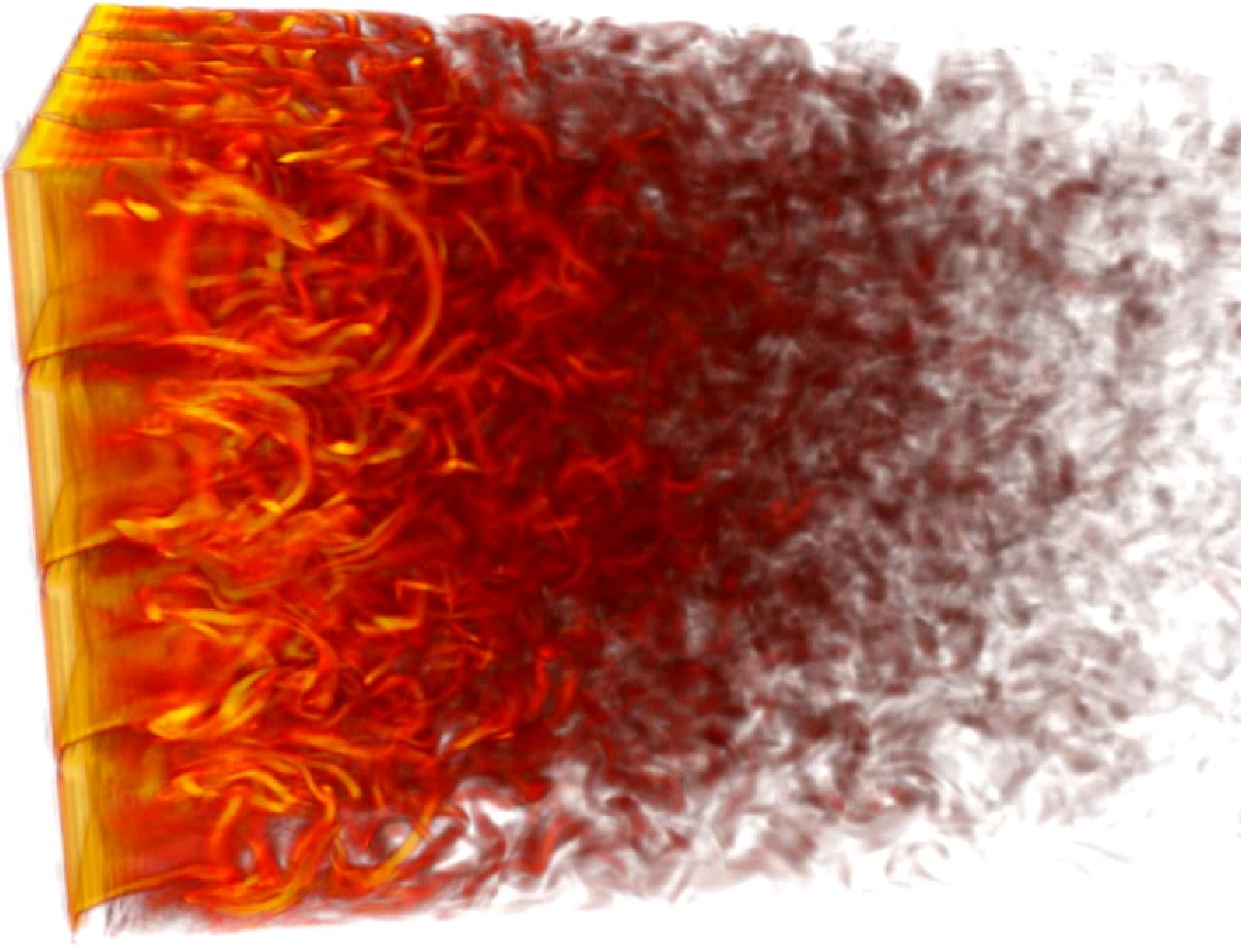
$$\tilde{\tilde{\mathbf{u}}}^{(n+1)} = \mathcal{B}\tilde{\mathbf{u}}^{(n+1)} = \tilde{\mathbf{u}}^{(n+1)} + \mathbf{f}_h^n \ .$$

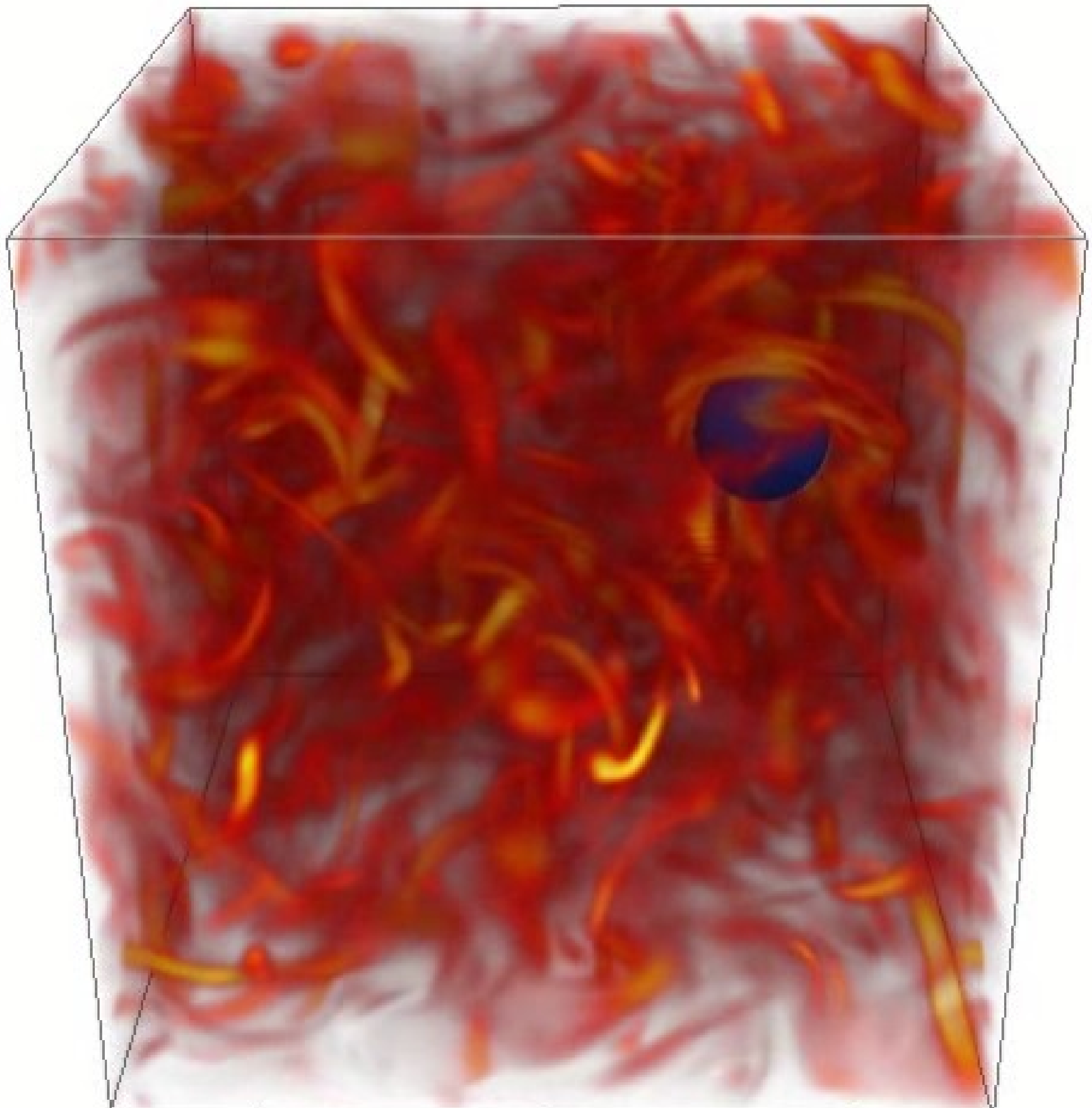
$$p^n = p^{n-1} + \Phi^n \ , \ \text{where} \ \Delta\Phi^n = \frac{1}{\Delta t}\nabla\cdot\tilde{\tilde{\mathbf{u}}}^{(n+1)} \ .$$

$$\mathbf{u}^{(n+1)} = \mathcal{P}\tilde{\tilde{\mathbf{u}}}^{(n+1)} = \tilde{\tilde{\mathbf{u}}}^{(n+1)} - \nabla p^n$$

H. Homann, J. Bec & R. Grauer JFM 2013.
Effect of turbulent fluctuations on the drag and lift forces on a towed sphere and its boundary layer







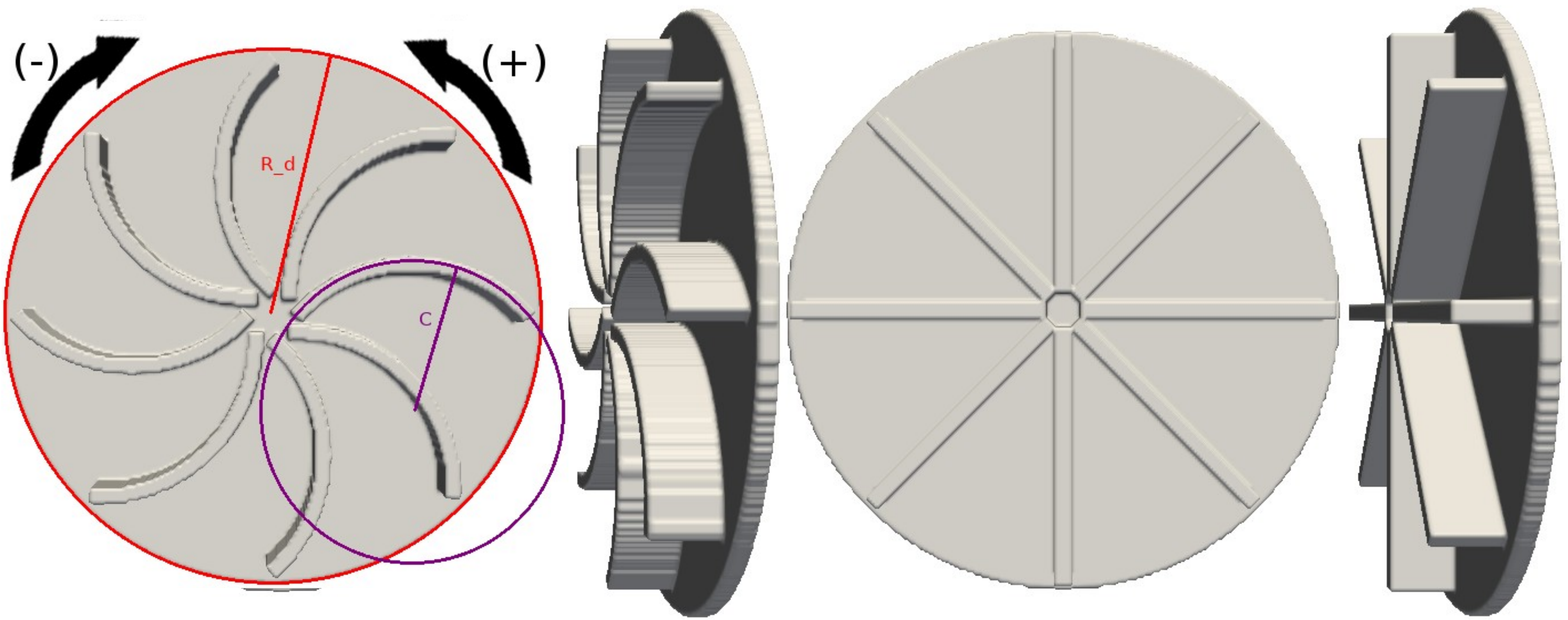
Design of the numerical impellers :

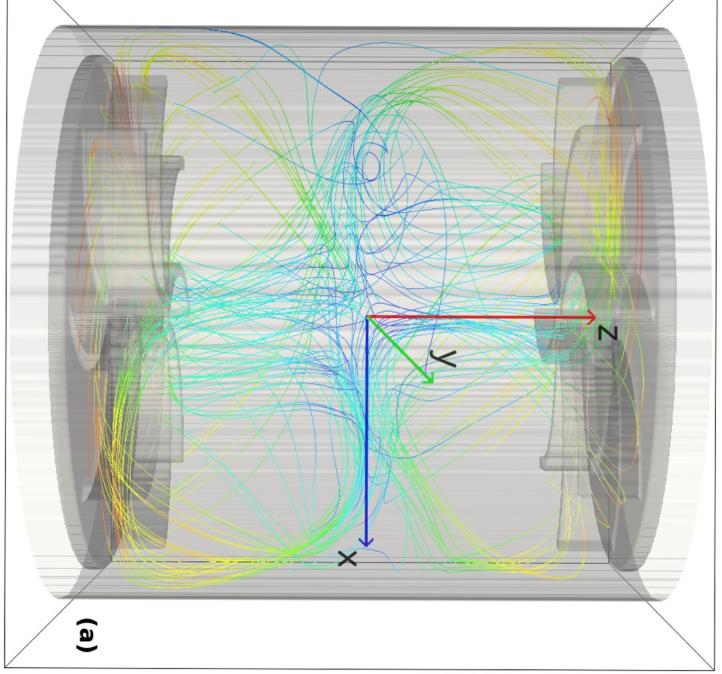
L. Marie et al   EPJB 33(4):469-485, June 2003.

TM28 configuration :

Rc = 3:0 , Rd = 0:9Rc, C = 0:5Rc
height of the eight blades is fix at 0:2Rc

Expulsion angle α = arcsin (Rd=2C) ~ 1:11976 rad ~ 64:15 deg.

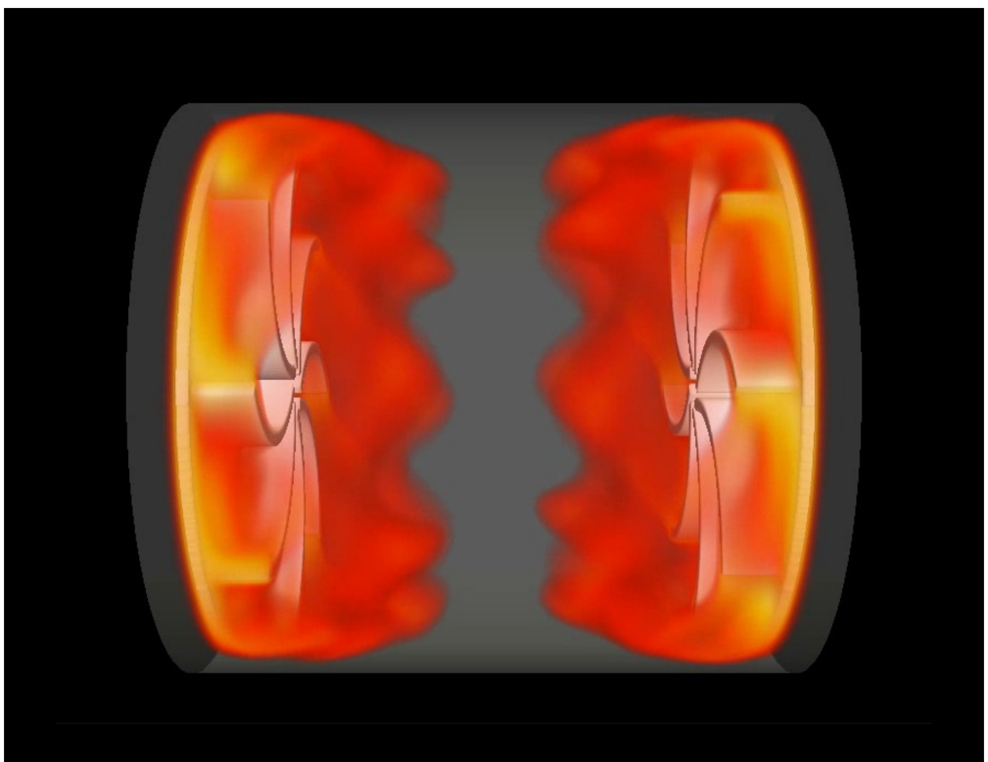




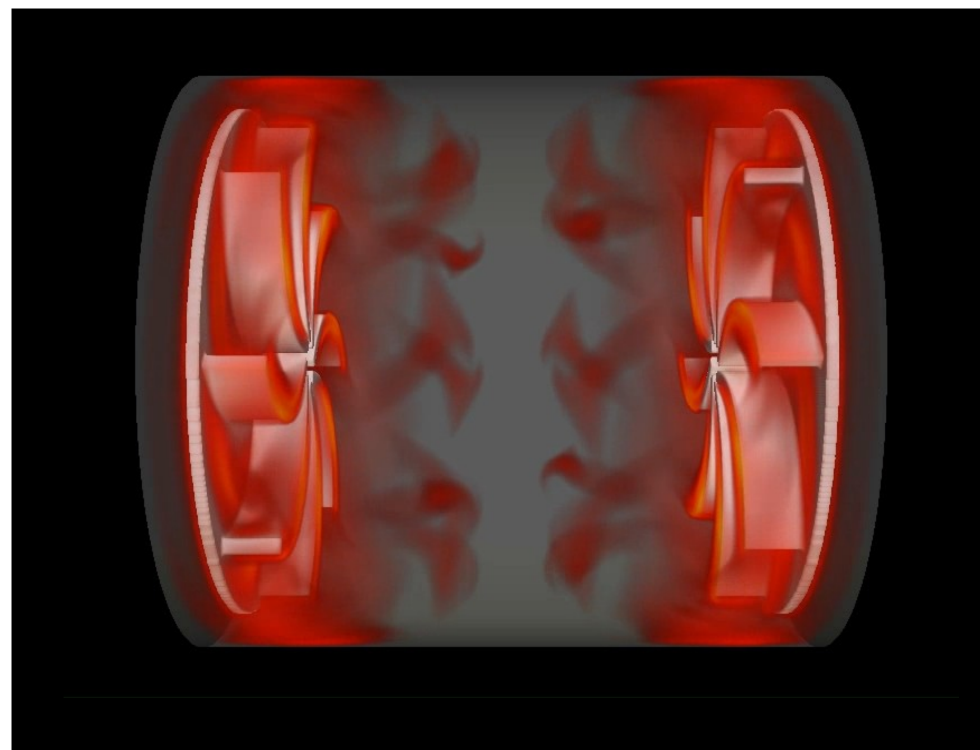
S. Kreuzahler, D. Schulz, H. Homann, Y. Ponty, R. Grauer

"Numerical study of impeller-driven von Karman flows via a volume penalization method"

New J. Phys. 16 103001 (2014)
doi:10.1088/1367-2630/16/10/103001

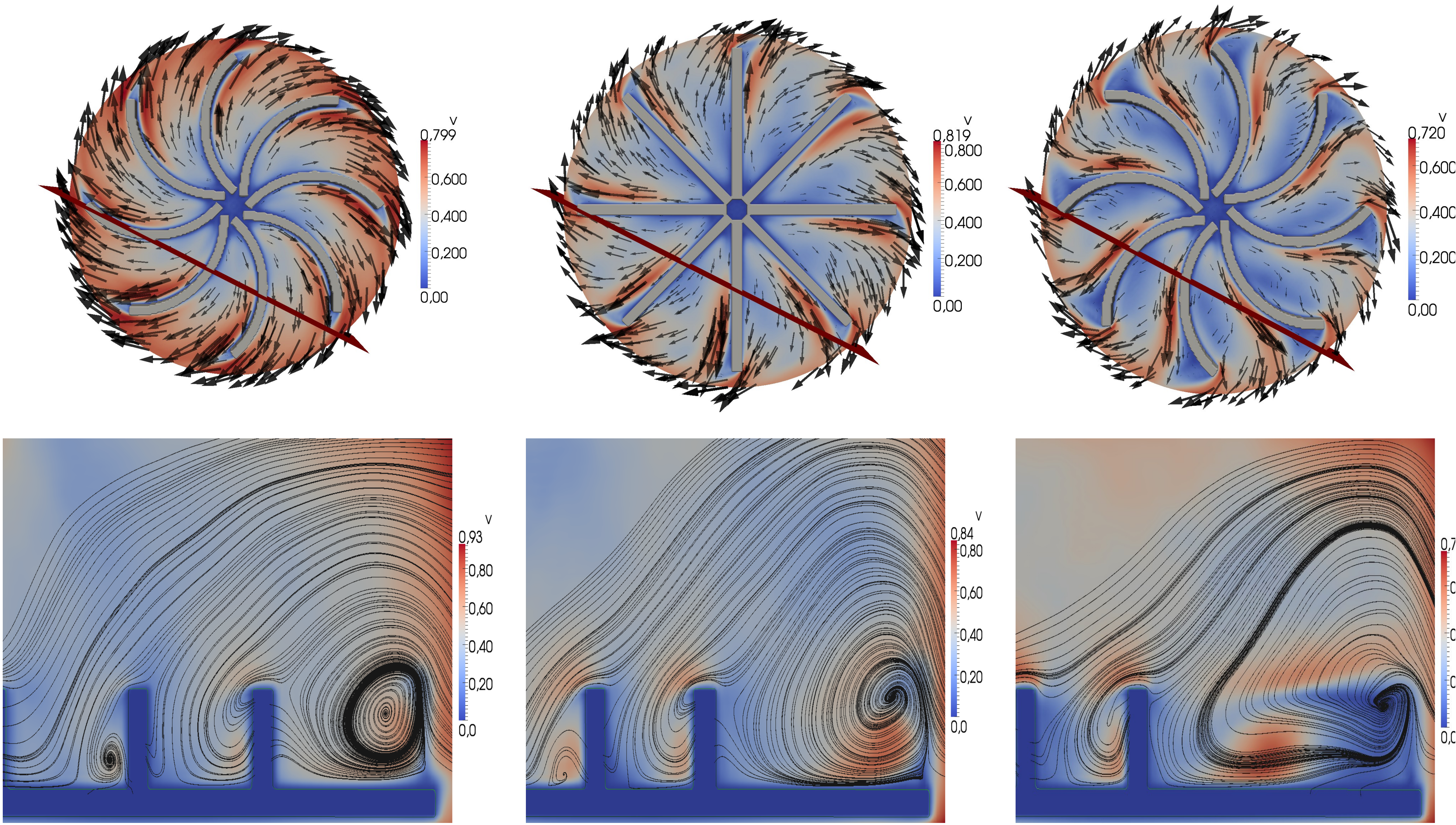


Kinetic Energy

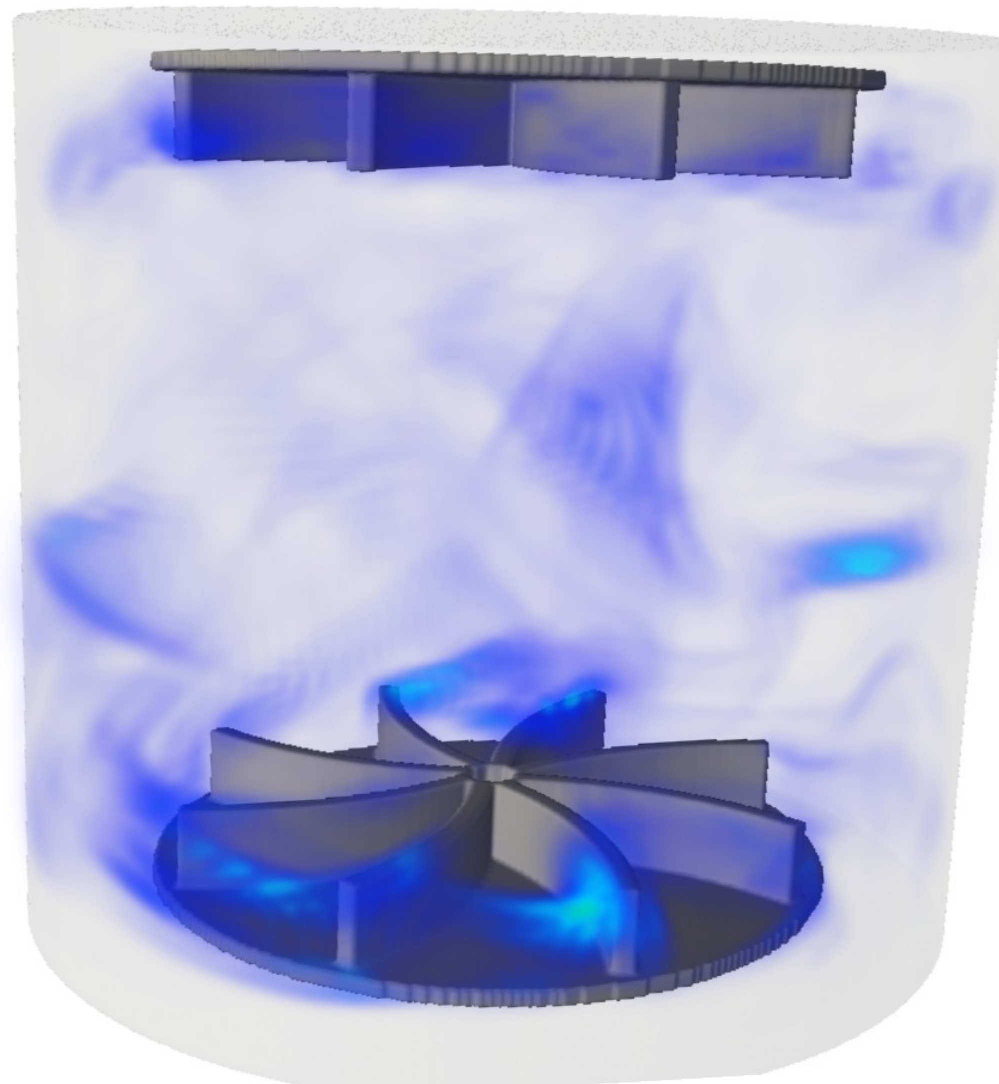


Enstrophy

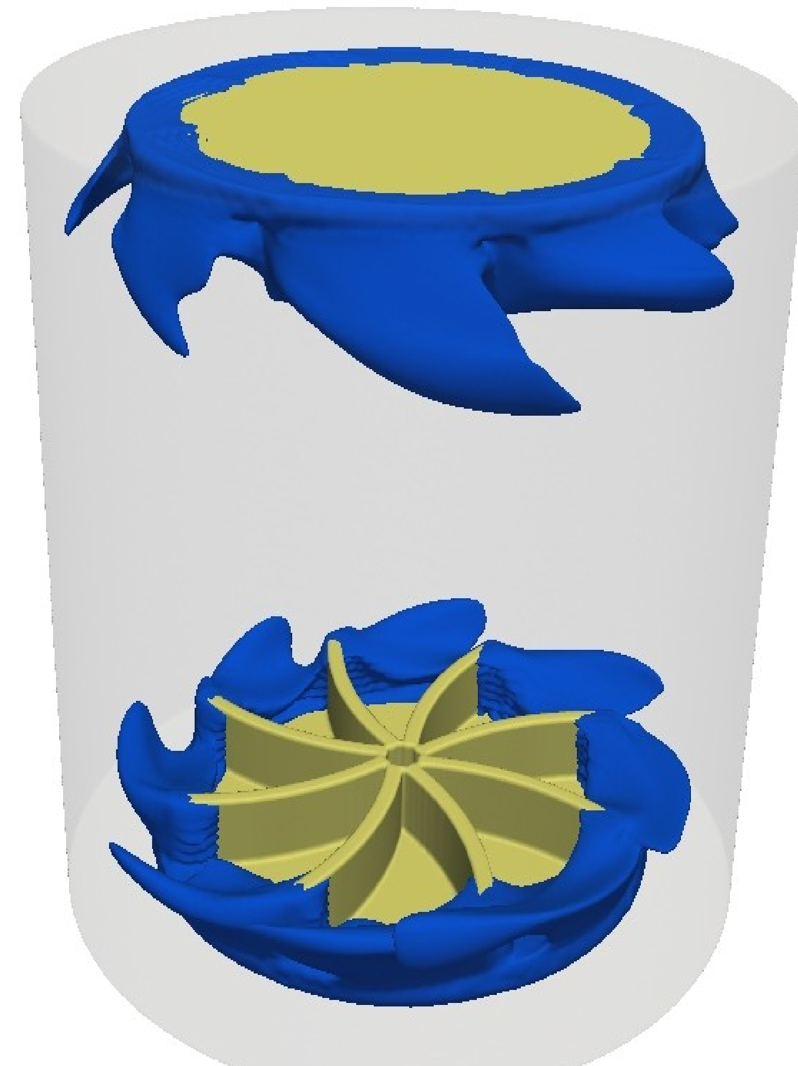
(+) , straight, (-) configurations



Sebastian Kreuzahler, Daniel Schulz, Holger Homann, Yannick Ponty, Rainer Grauer
"Numerical study of impeller-driven von Karman flows via a volume penalization method"
New J. Phys. 16 103001 (2014)

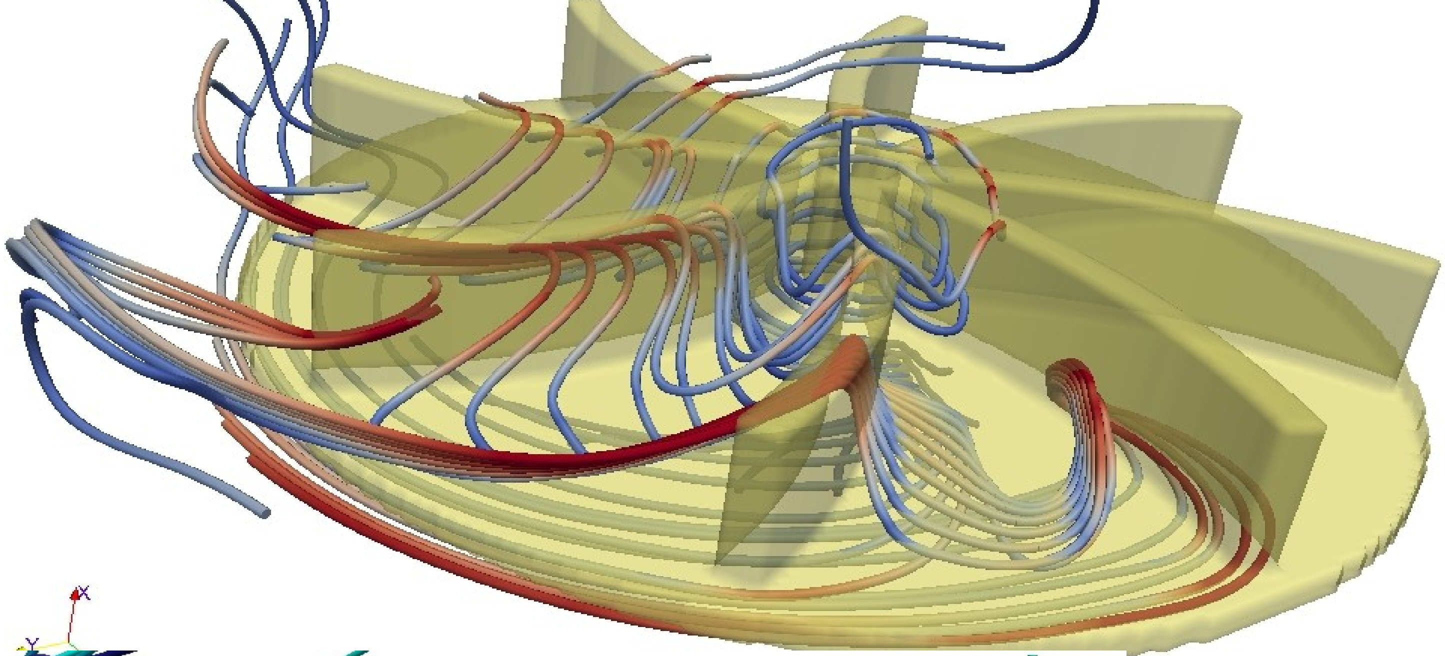


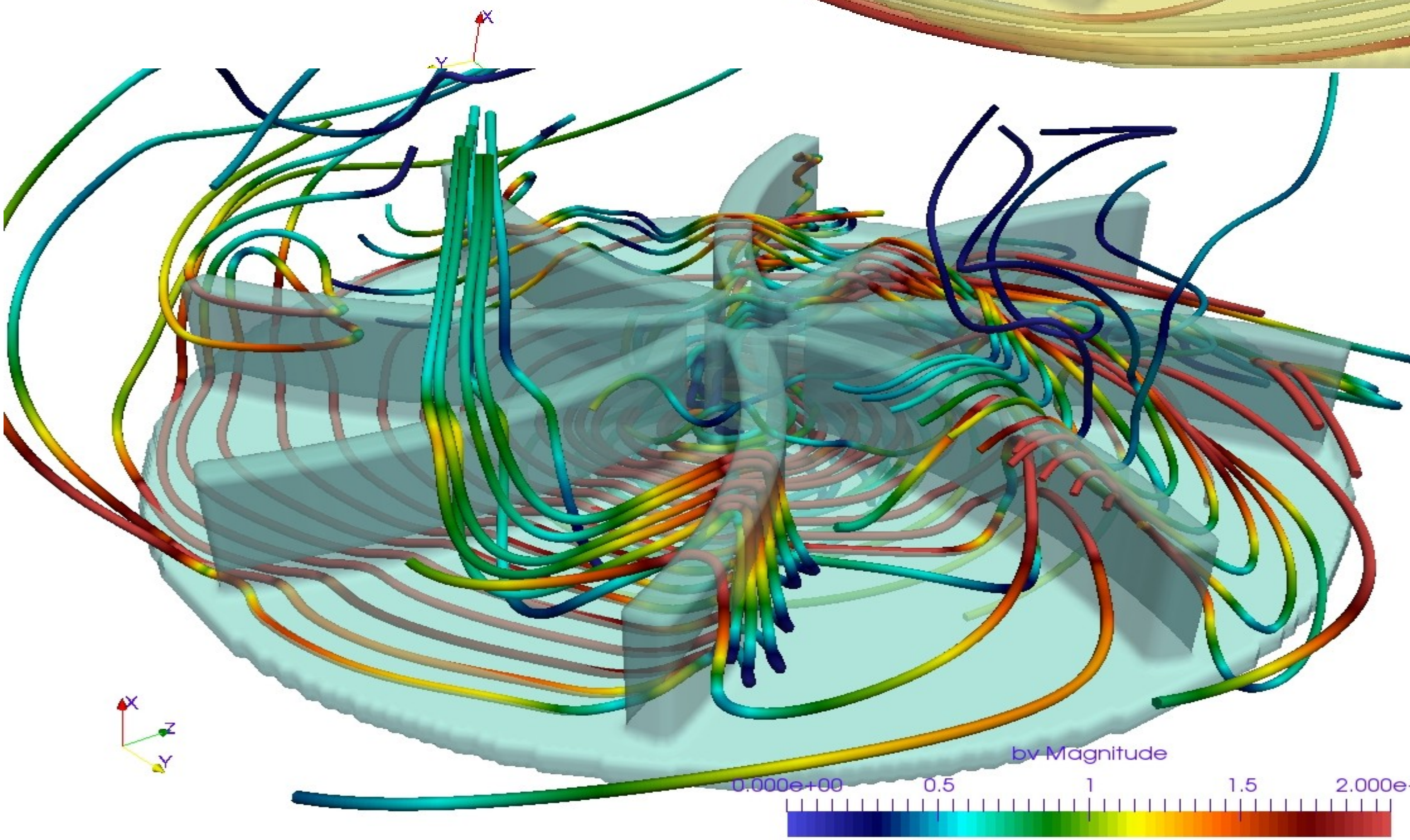
Enstrophy



Kinetic energy

Dynamo





Pseudo-Penalization method

Near the boundaries the schemes is accurate only order h
-> no spectral accuracy near the boundaries

Lost of grid point

Easy to implement , versatile
-> EASY boundaries

Prospective numerical tool

What do you need to built a nice

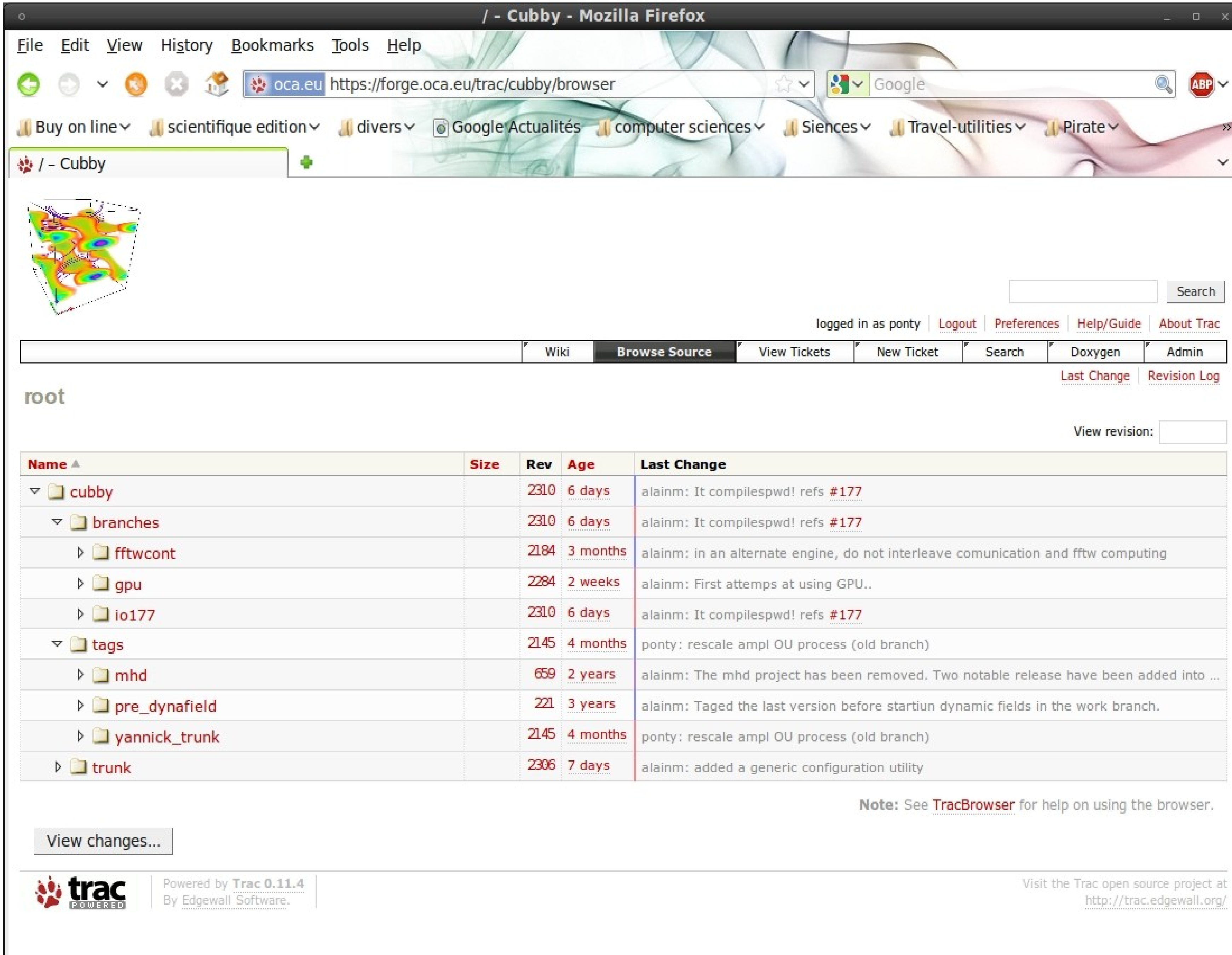
Editor → emacs, debugger (db)
Language → c++ , F90
Clusters → ask to your boss or institutes

- Version saver → subversion
- Wiki, ticket bug → trac

Postraitement :
3D Visualisation → paraview, vapor
1D, 2D plots :
Matlab → ask to your boss

Python : matplotlib , scipy, ….

Subversion (svn) + trac



Changeset 2299 for cubby/trunk/main.cpp - Cubby - Mozilla Firefox

File Edit View History Bookmarks Tools Help

oca.eu https://forge.oca.eu/trac/cubby/changeset/2299/cubby/trunk/ Google

Buy on line scientifique edition divers Google Actualités computer sciences Siences Travel-utilities Pirate

Changeset 2299 for cubby/trun...

```
                rv(avg_pmagn_d.Jms(space)) = 0;
            }

            if (avg_pfluid::on()) {
                avg_pfluid::data_type& avg_pfluid_d = avg_pfluid::data();
                rv(avg_pfluid_d.Psims(space))= 0;
                rv(avg_pfluid_d.Wms(space)) =0;
            }
            if (avg_hely::on()) {
                avg_hely::data_type& avg_hely_d = avg_hely::data();
                rv(avg_hely_d.Hm(space)) = 0;
                rv(avg_hely_d.Hms(space)) = 0;
            }
            if (avg_pfluid::on()) {
                avg_pfluid::data_type& avg_pfluid_d = avg_pfluid::data();
                rv(avg_pfluid_d.Psims(space))= 0;
                rv(avg_pfluid_d.Wms(space)) =0;
            }
            if (avg_hely::on()) {
                avg_hely::data_type& avg_hely_d = avg_hely::data();
                rv(avg_hely_d.Hm(space)) = 0;
                rv(avg_hely_d.Hms(space)) = 0;
            }

        }

        // - - - -  Penalization - - - - - - - - - - - - - -

        if (params.physical.penalization_v) {
            field::scalar P_xi(space, "penalization", field::sp_real);
            physic::make_penalization_field( P_xi,
                                             params.physical.penalization_v,
                                             params.physical.radius,
                                             params.physical.DH);
            io_oengine->store( P_xi, 1.0, 1.0, cst_name( "P_xi.dat" ) );
            io_oengine->store( P_xi, 1.0, 1.0, "P_xi.dat" );
        }
        if (params.physical.penalization_f) {
            field::scalar P_xi(space, "penalization", field::sp_real);
            physic::make_penalization_field( P_xi,
                                             params.physical.penalization_f,
                                             params.physical.radius_f,
                                             params.physical.DH_f);
            io_oengine->store( P_xi, 1.0, 1.0, cst_name("P_force.dat") );
            io_oengine->store( P_xi, 1.0, 1.0, "P_force.dat" );
        }

        params.fluid.nu_eff = (params.physical.turbulent_viscosity > 1
                               ? params.fluid.nu + params.fluid.nu_min
                               : params.fluid.nu );

        out << "After initialisation of v : nu = " << std::scientific << params.fluid.nu
            << "  nu_eff=" << std::scientific << params.fluid.nu_eff
            << "  nu_min=" << std::scientific << params.fluid.nu_min << std::endl;

```

Done

Python libraries are great and free !

Scientific python libraries: scypy

Matplotlib : plot 1D, 2D

Glue language : one python script could mix :
"linux command",
executable code, …

Interpreted language → no compilation
Some part could be compile automaticaly
Or push to be compile ()

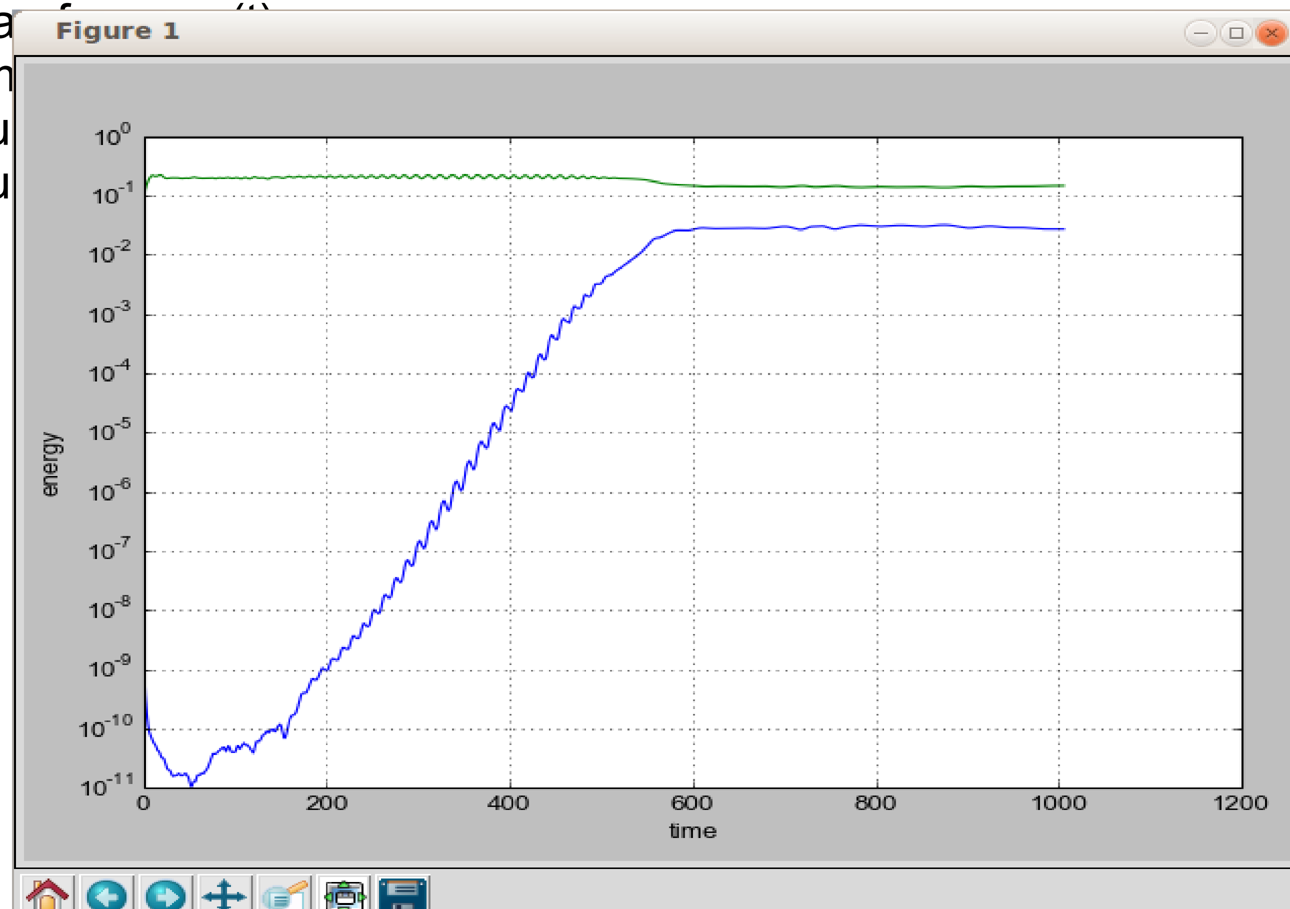
Parallel script : multiprocess, mpi, gpu

Parser option

```
#!/usr/bin/env python
from pylab import *
from optparse import OptionParser
import string
from scipy import stats

####### Manage the options :

parser = OptionParser()
parser.add_option('-v', '--inputv', action="store", type="string",\
                dest="inputv", default='energy.log',\
                help='input velocity energy file (default:energ_v.dat)')
parser.add_option('-l', '--linb', action='store_true',\
                dest='linb', default=False, \
                help='linear y axis for magnetic plot (log instead)')
parser.add_option('-t', '--total', action='store_true',\
                dest='total', default=False, \
                help='Plot total energy (kinetic + magnetic)')
….
```

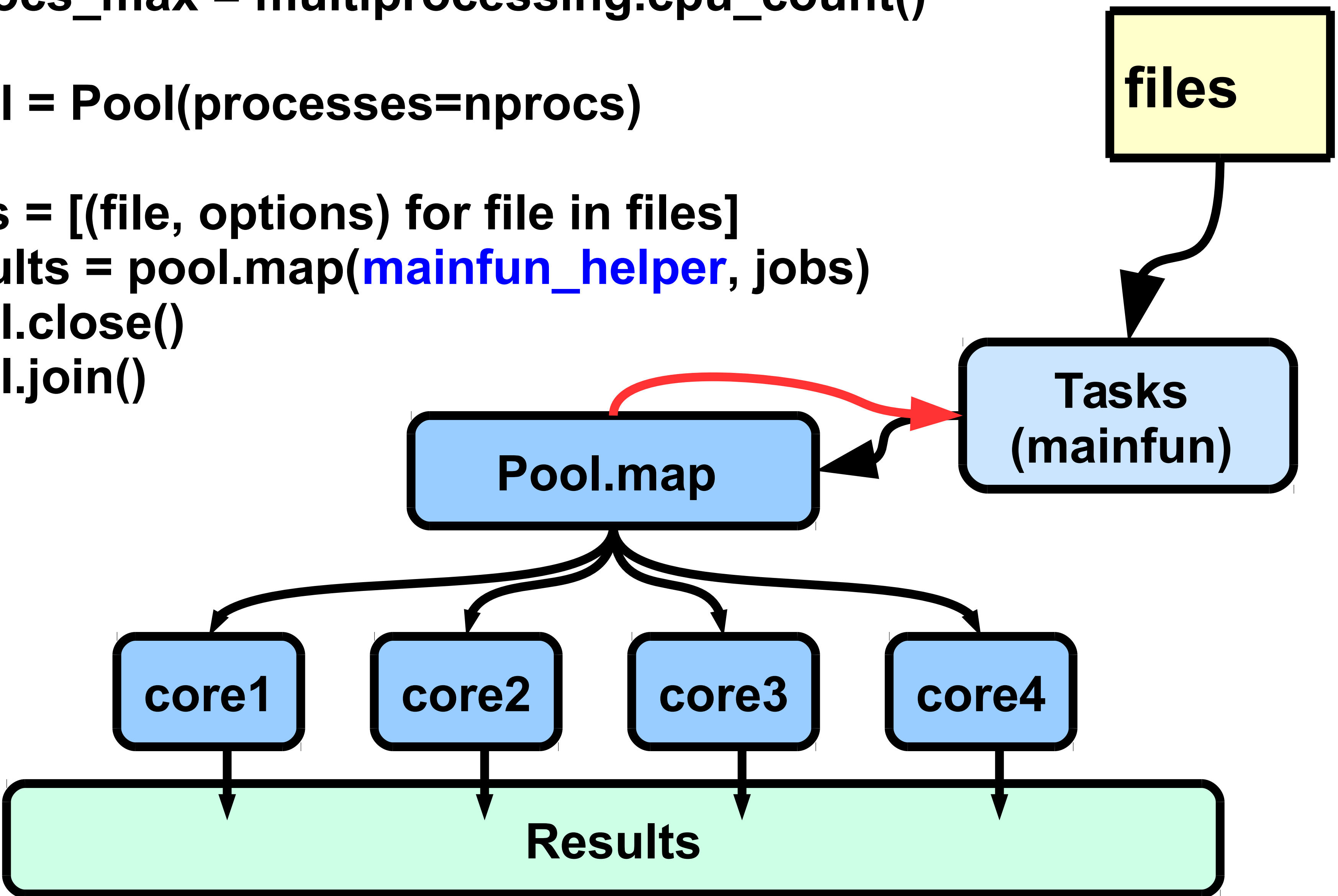
```
ponty@hobbit:~$ plot.py --help
Usage: plot.py [options]

Options:
  -h, --help            show this help message and exit
  -v INPUTV, --inputv=INPUTV
                        input velocity energy file (default:energ_v.dat)
  -l, --linb            linear y axis for magnetic plot (log instead)
  -t, --total           Plot total energy (kinetic + magnetic)
  --together            Plot total energy (kinetic + magnetic) together
  -p, --pdf             print all the graphic directly inside pdf file
  -o, --pdf_only        print all the graphic directly inside pdf file only
  -s, --spectra         Plot the spectra
  --stat                Plot the histogram
  --five                input has five colu
  --step                plot energy versu
```

plot.py -v energy.log



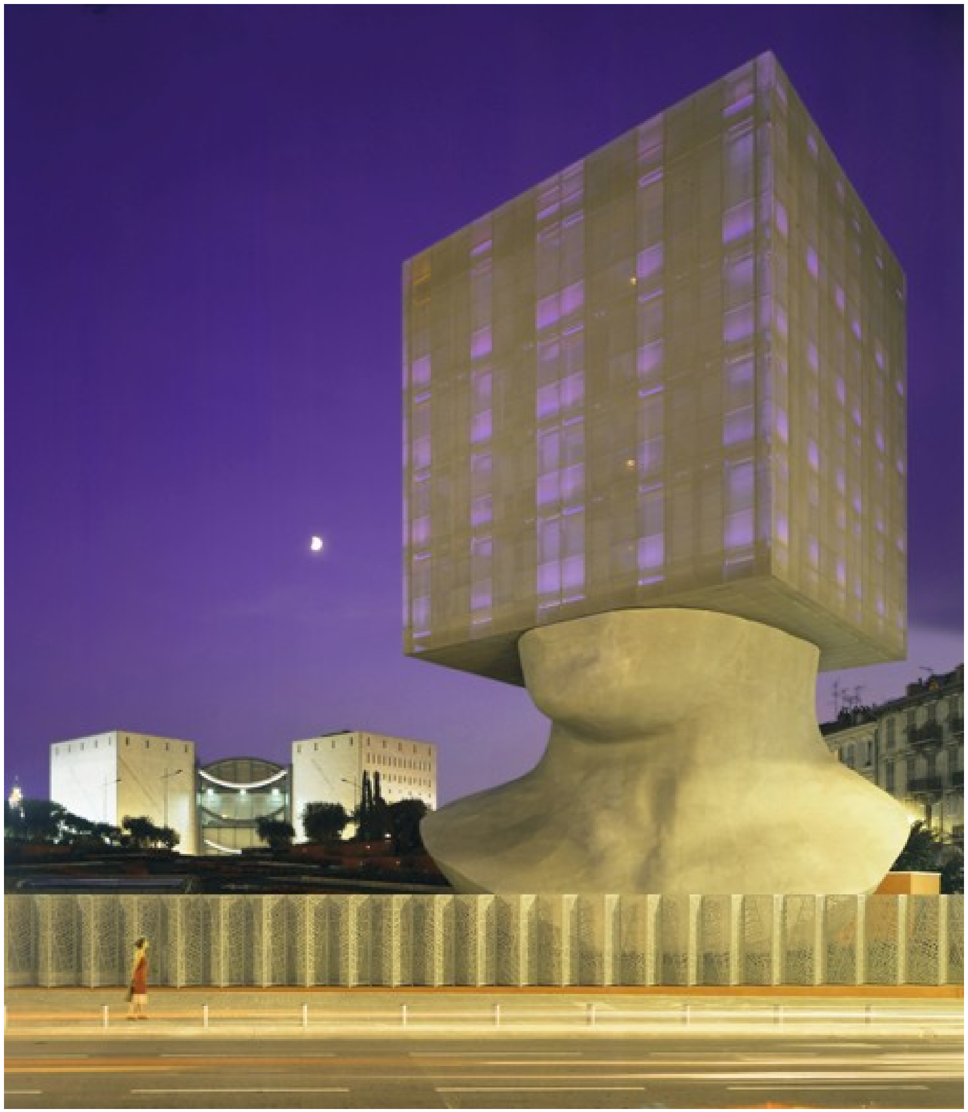
Parallel task with Python

```
from multiprocessing import Pool
import multiprocessing

        nprocs_max = multiprocessing.cpu_count()
        …
        pool = Pool(processes=nprocs)

        jobs = [(file, options) for file in files]
        results = pool.map(mainfun_helper, jobs)
        pool.close()
        pool.join()
```



Impossible to escape
Then Join the

Thank Annick to put me inside this bloody CUBE !!!



Scientific CUBE institute at Nice