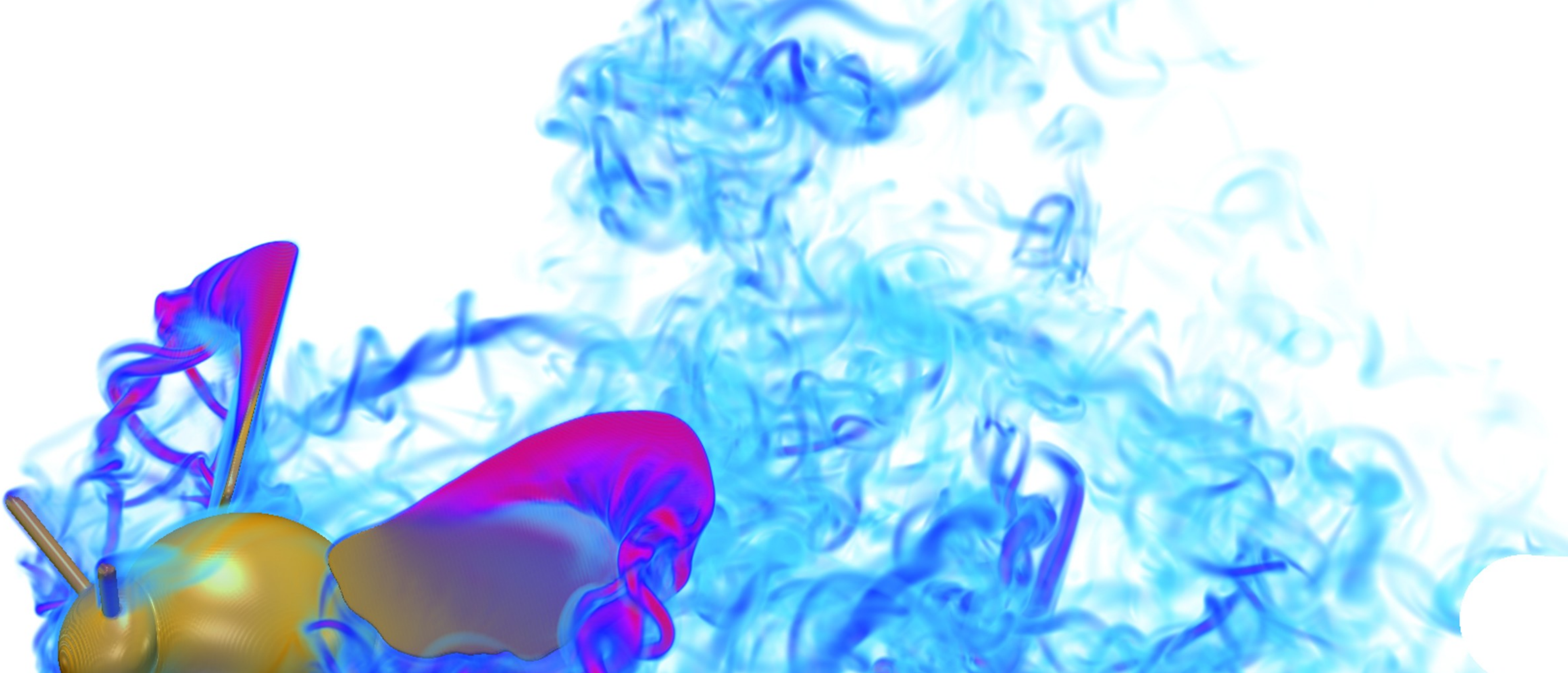
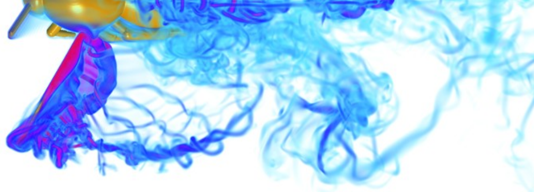
# Progress on adaptive insect simulations

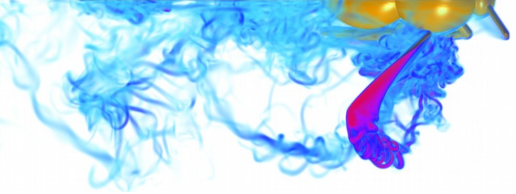

**Thomas Engels**

École Normale Supérieure Paris,
Laboratoire de Météorologie Dynamique


Internal presentation at the 5th International AIFIT workshop







# Looking back on Fourier

- Until now, we used the FluSI code, which is based on Fourier transforms
- Very accurate, fast, massively parallel allows fast inversion of Laplace operator → perfectly incompressible
- No boundary conditions: Volume penalization method

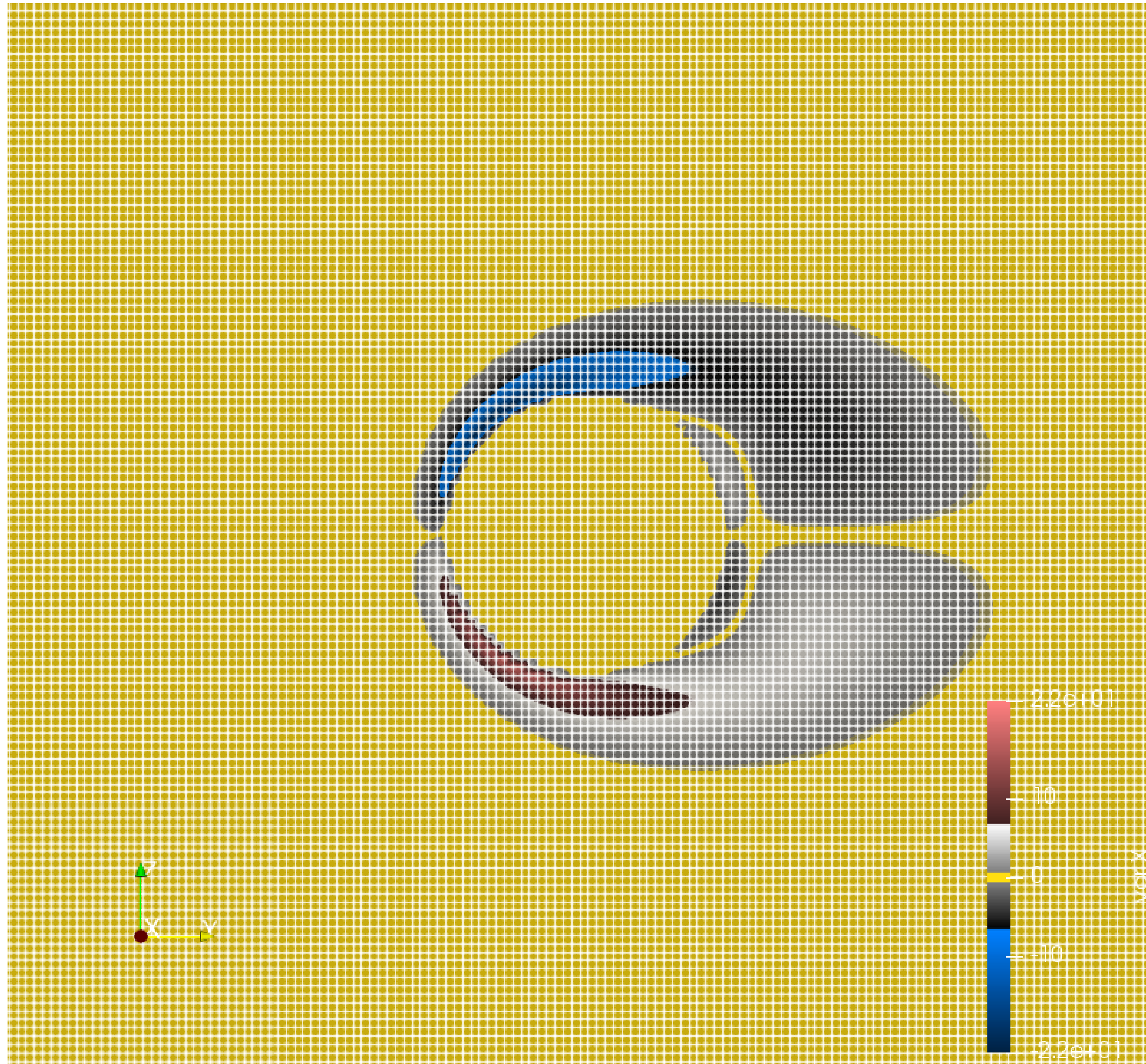
$$\widehat{u}_k = \frac{1}{N}\sum_{n=0}^{N-1} u_n \cdot e^{-i2\pi kn/N}$$

$$u = \sum_{k=0}^{N-1} \widehat{u}_k \cdot e^{i2\pi kn/N}$$

$$\widehat{(\partial_x u)} = ik\widehat{u}_k$$

# Looking back on Fourier

- Equidistant grids: resolution is the same everywhere
- Limits domain size or resolution

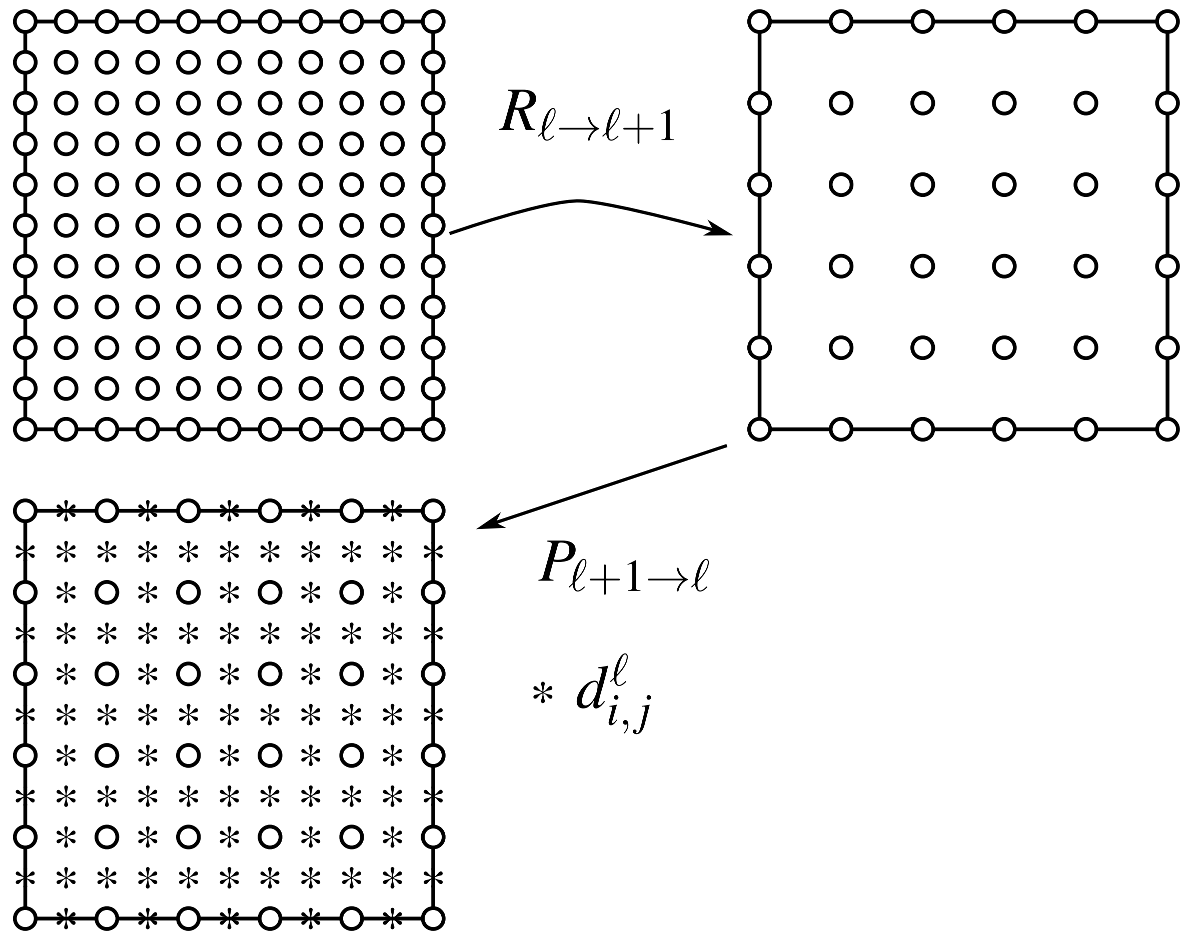


# The basic idea for adaptivity

- Refine where necessary, coarsen where possible
- Two groups of methods appear: error *indicated* and error *controlled*
- Error indicators: (adaptive mesh refinement)
  - Vorticity, strain rates, flame positions etc
- Error controlled:
  - Multiresolution methods, many ideas from A. Harten in 1990s
  - Equivalent to bi-orthogonal wavelets
- Bye bye Fourier

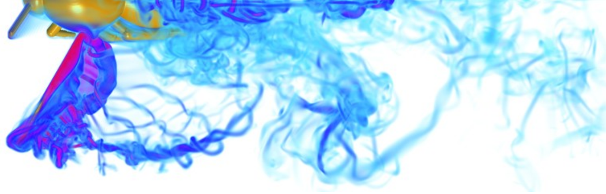
# Computation of details

- Detail coefficients are obtained by coarsening, then refining
- Their magnitude is related to the local regularity of the solution
- Small details in smooth regions → adaptive coarsening
- Large details → resolution is required.

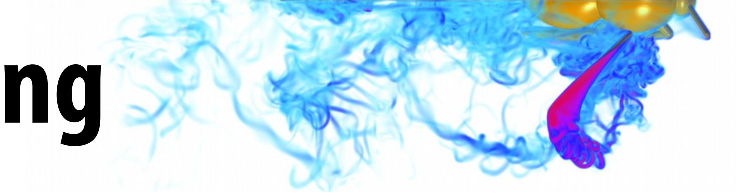


# The multiresolution algorithm

- To advance the solution from $t^n$ to $t^{n+1}$
  - Refinement stage. The entire grid is refined, to be sure that it is sufficient to contain the solution at the *new* time level
  - Evolution. Solve the PDE using finite differences and advance in time
  - Coarsening. Check the details and keep only those points where the details are significant.
  - Load balancing. The number of points has changed, and on some CPU we might now have more points. This must be corrected.

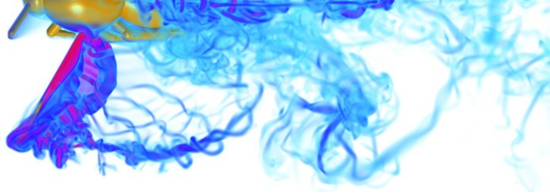


# The error balancing

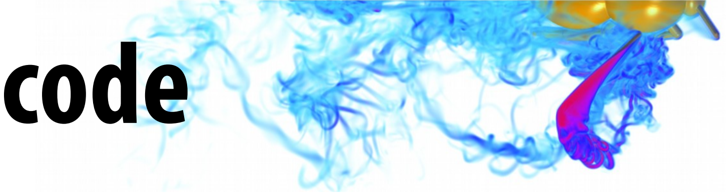


- Idea is to balance error from the compression and discretization

$$|u_N^\varepsilon - u^{ex}| \leq \underbrace{|u_N - u_N^\varepsilon|}_{\text{Perturbation error } \mathcal{O}(\varepsilon) N_t} + \underbrace{|u_N - u^{ex}|}_{\text{discretization error}}$$



# Main ideas of new code



- Sacrifice strict incompressibility for efficiency (and hopefully overall error reduction)

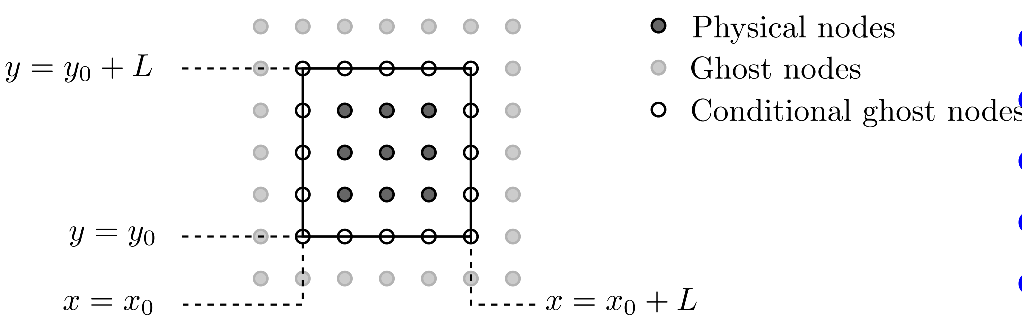
$$\partial_t \underline{u} = -\left(\underline{u} \cdot \nabla\right) \underline{u} - \nabla p + \nu \nabla^2 \underline{u} - \frac{\chi}{C_\eta}\left(\underline{u} - \underline{u}_s\right) - \frac{\chi_{\mathrm{sp}}}{C_{\mathrm{sp}}}\left(\underline{u} - \underline{u}_\infty\right) \quad (1)$$

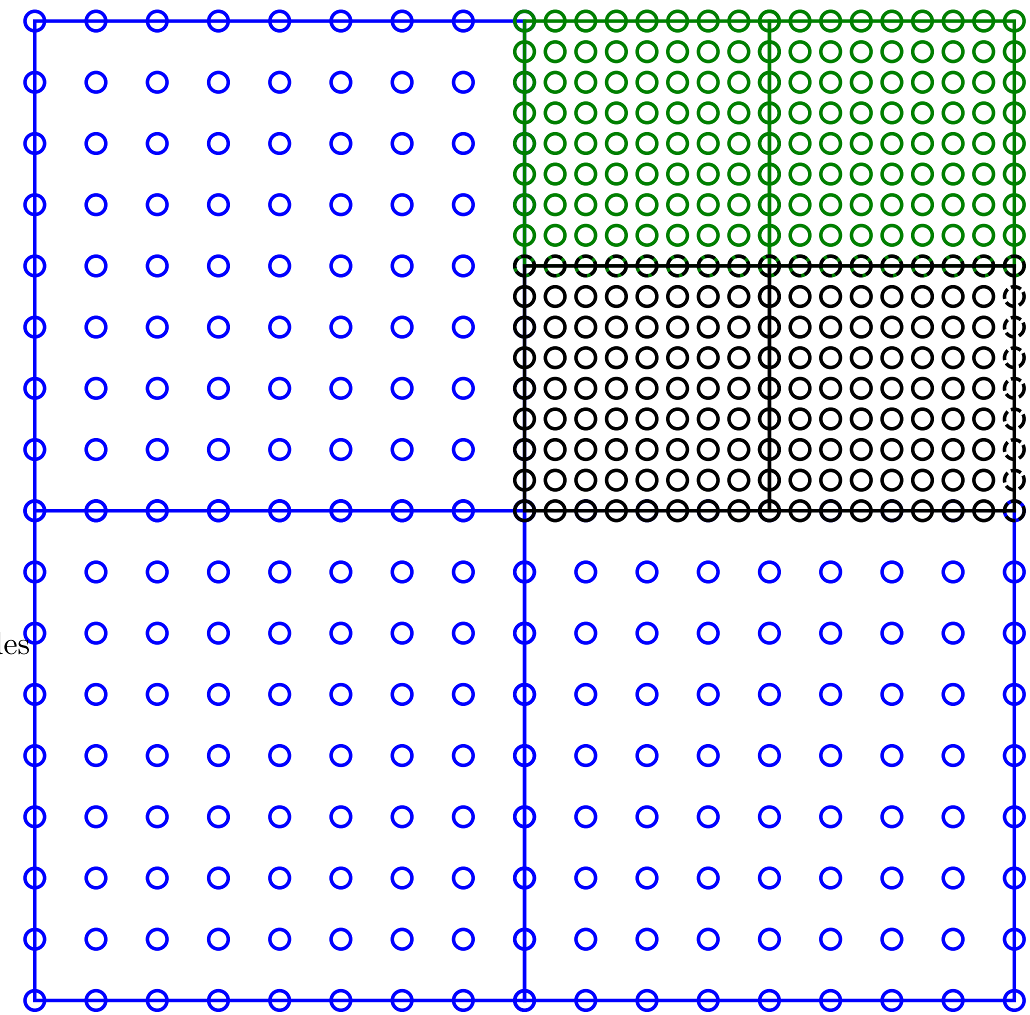
$$\partial_t p = -c_0^2 \nabla \cdot \underline{u} - \gamma p - \frac{\chi_{\mathrm{sp}}}{C_{\mathrm{sp}}}\left(p - p_\infty\right). \quad (2)$$

- Sacrifice a part of compression in favor of faster data structures → block structured data (hybrid datastructure)
- Open source, multi-physics
- Wabbit: (W)avelet (A)daptive (B)lock-(B)ased solver for (I)nsects in (T)urbulence

# Block-structured grid approach

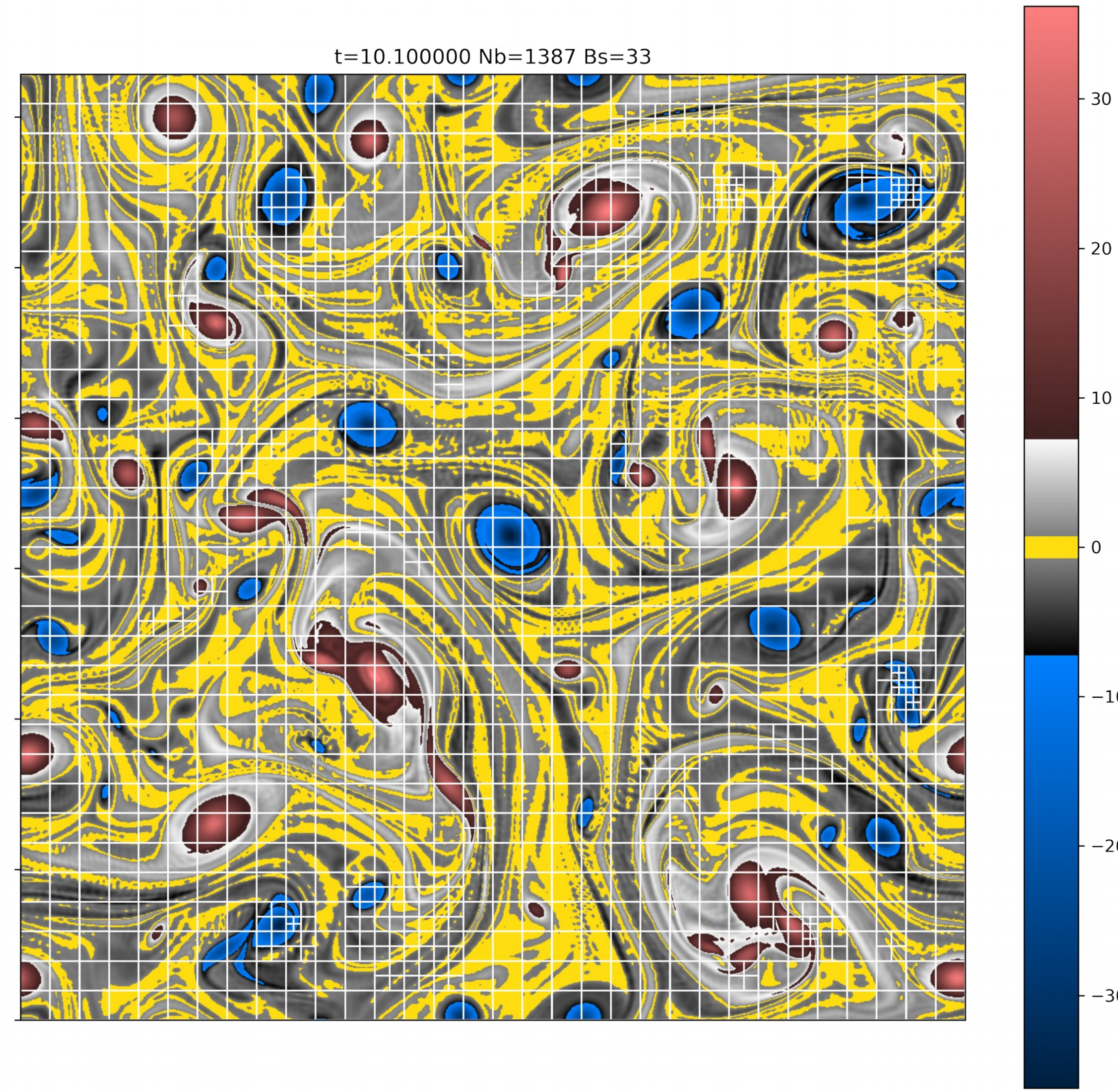
- Example grid with two levels and 7 blocks
- Each block has a layer of ghost nodes connecting them to neighboring blocks



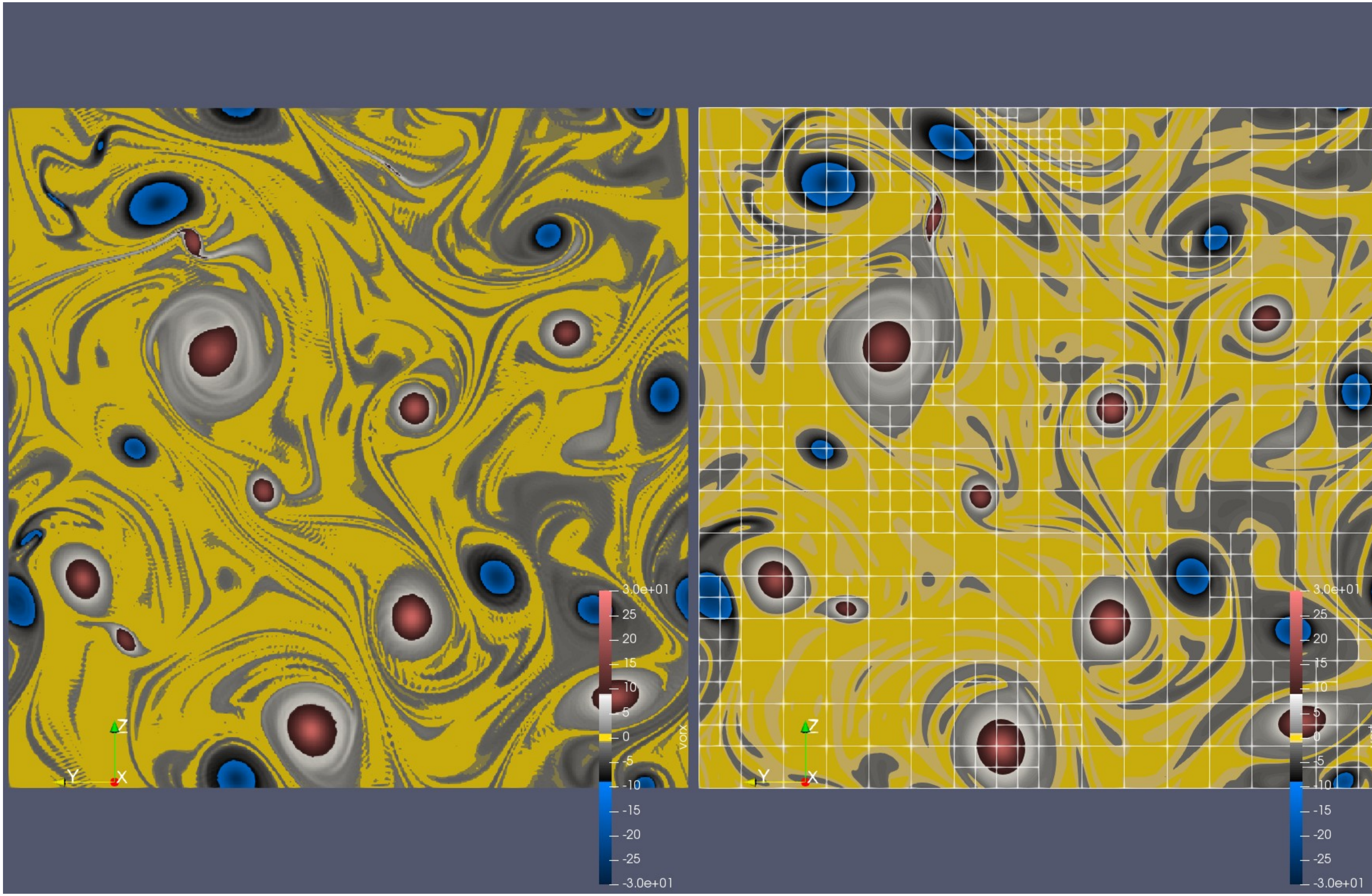



# Example I: decaying 2D turbulence

- In flusi: initialized random 2D vorticity field, let evolve Navier-Stokes until regularization has occurred, then use this field as initial condition for WABBIT.
- We compare evolution between flusi and WABBIT
- Note chaotic nature of problem, but just a first test.
- C0=50.0 eps=1e-3

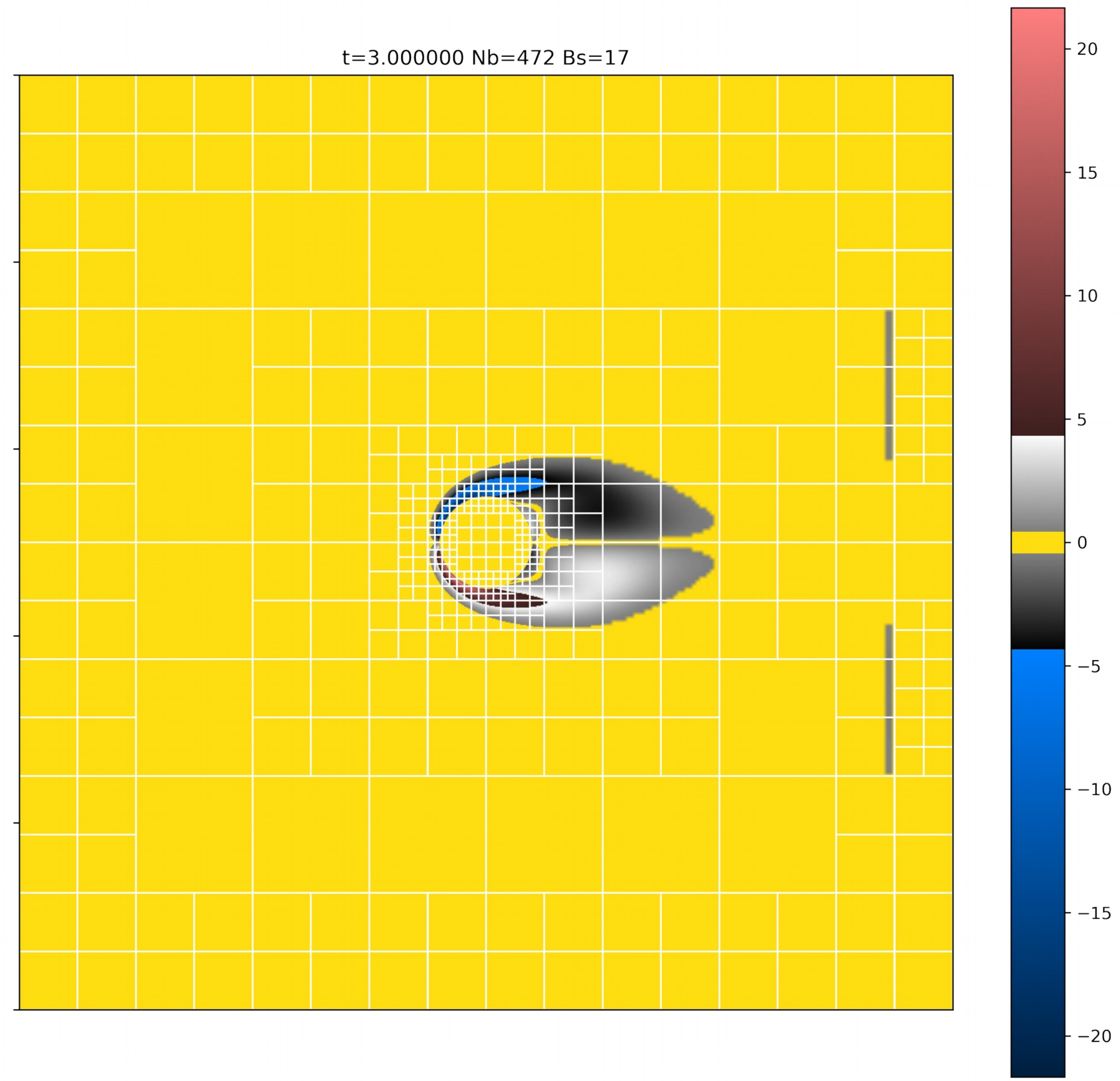


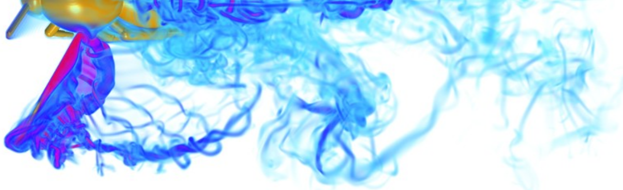
# Terminal fields



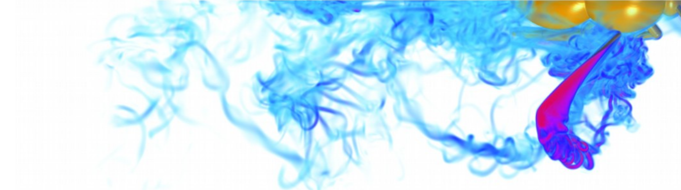
# Example II: flow past cylinder

- Impulsivey started flow around a cylinder at Re=100 c0=40 eps=1e-3 C_eta=1e-3
- Volume penalization interface on maximum level
- Non-reflecting outflow in penalization set



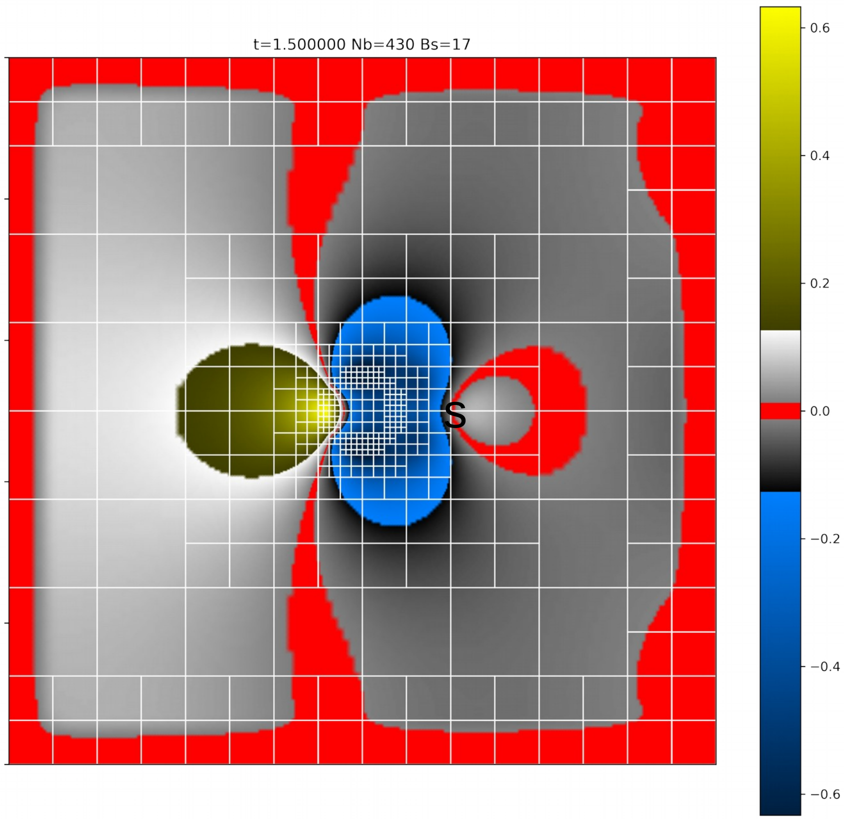


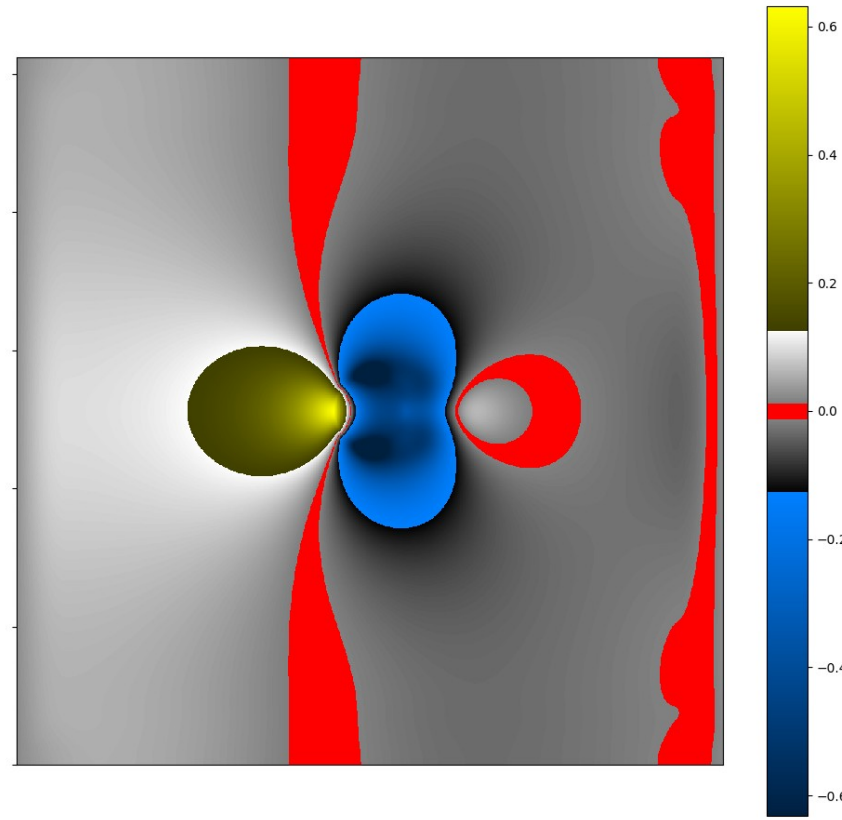
# force



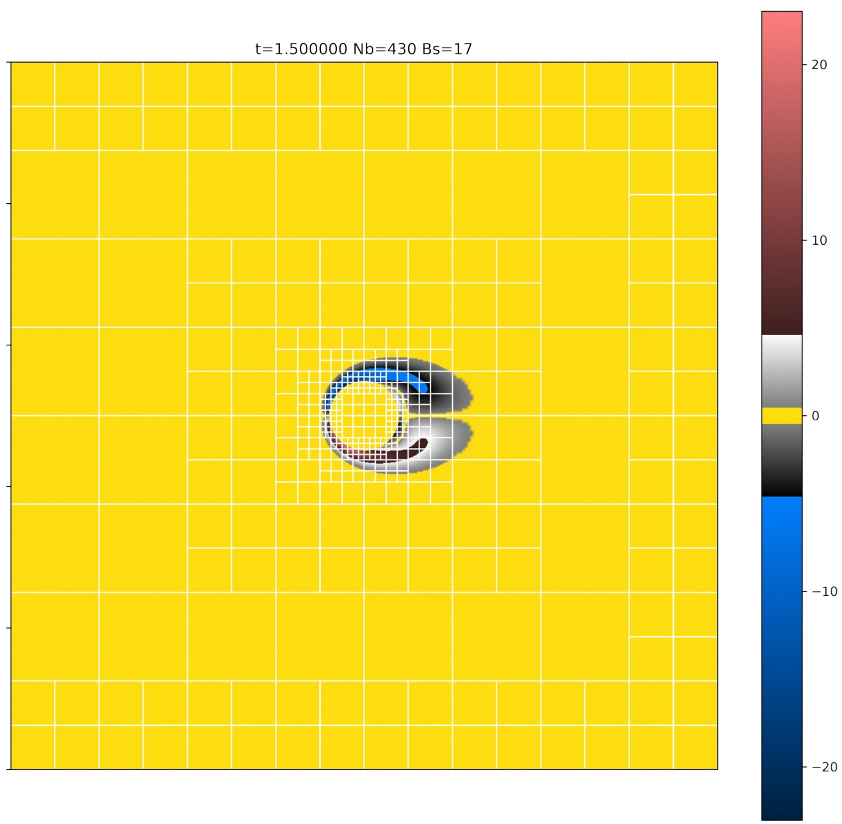
- Flusi Fx=8.265600e-01 Wabbit Fx=8.184966e-01
- Flusi: 1024^2

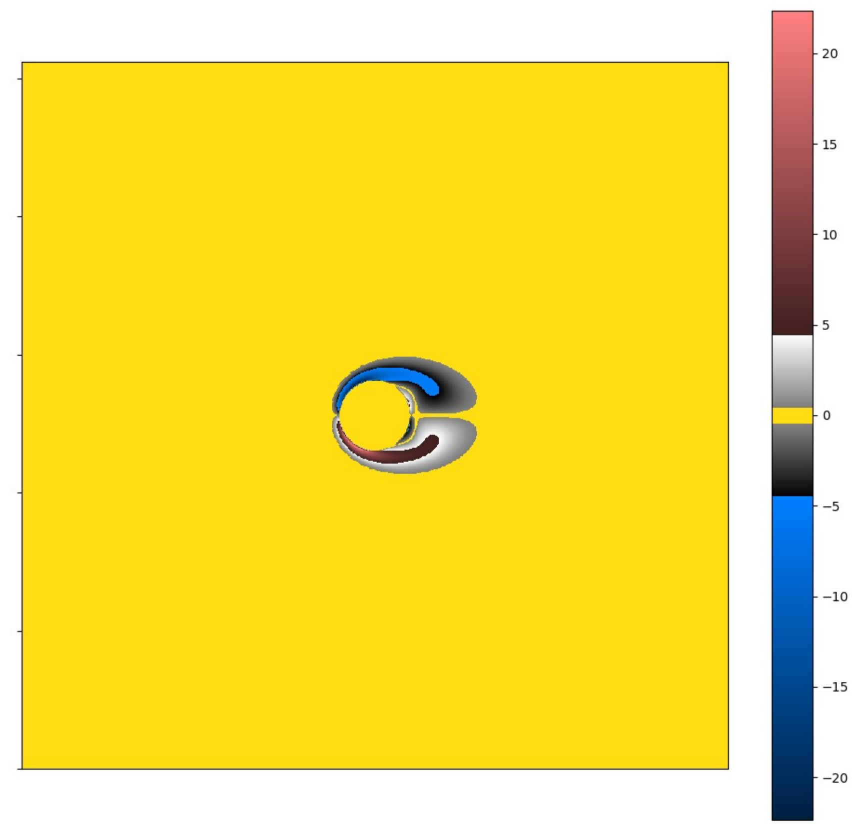
# Pressure at t=1.5



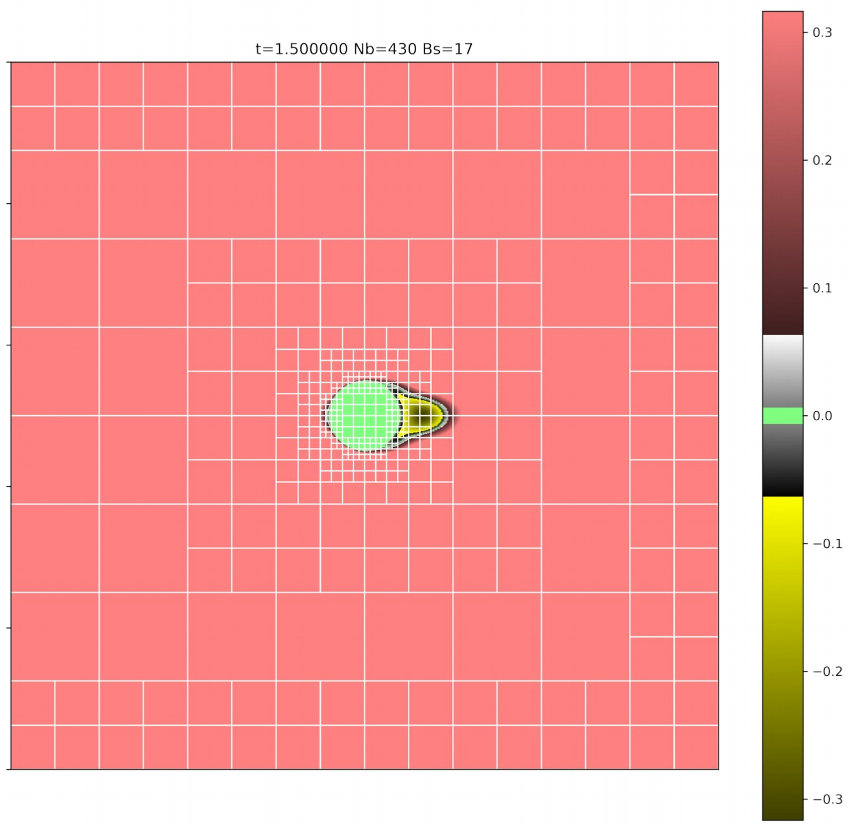



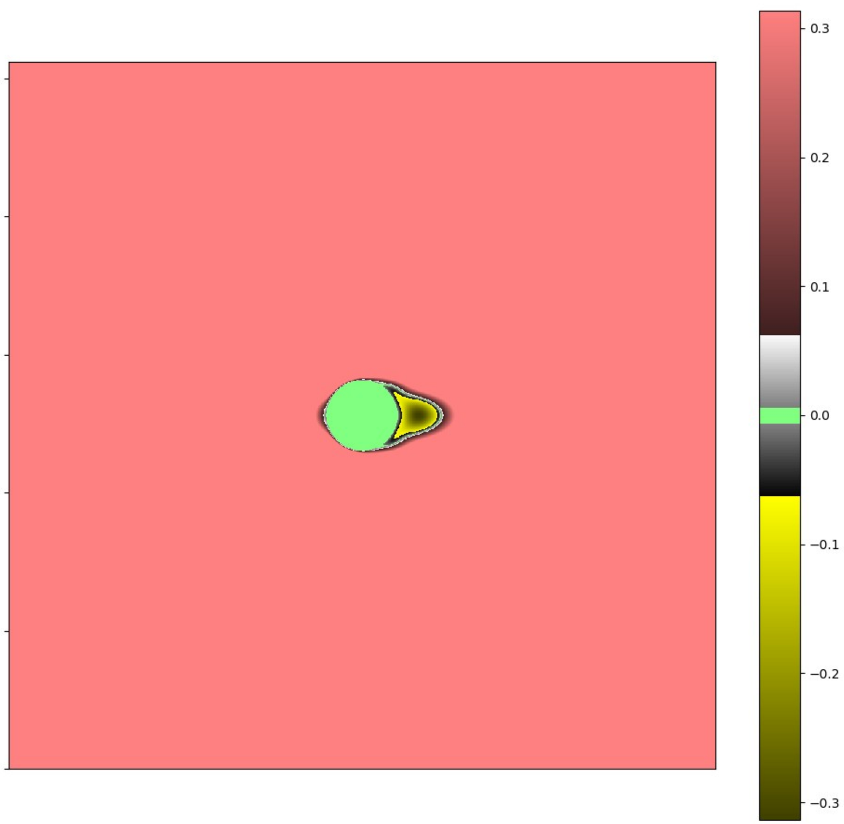
# vorticity

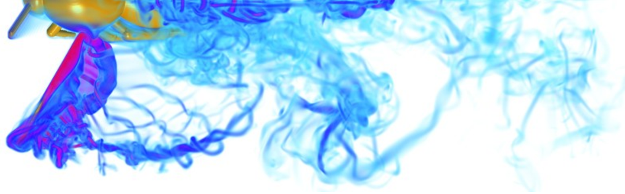


# Horizontal velocity





# Vertical velocity