A wavelet-adaptive method for multiscale simulation of turbulent flows in flying insects

Kai Schneider

I2M, Aix-Marseille Université, France

joint work with

Thomas Engels[1], Julius Reiss[1], Marie Farge[3]

[1]TU Berlin, Germany, [2]CNRS, ENS Paris, France

X WWLET

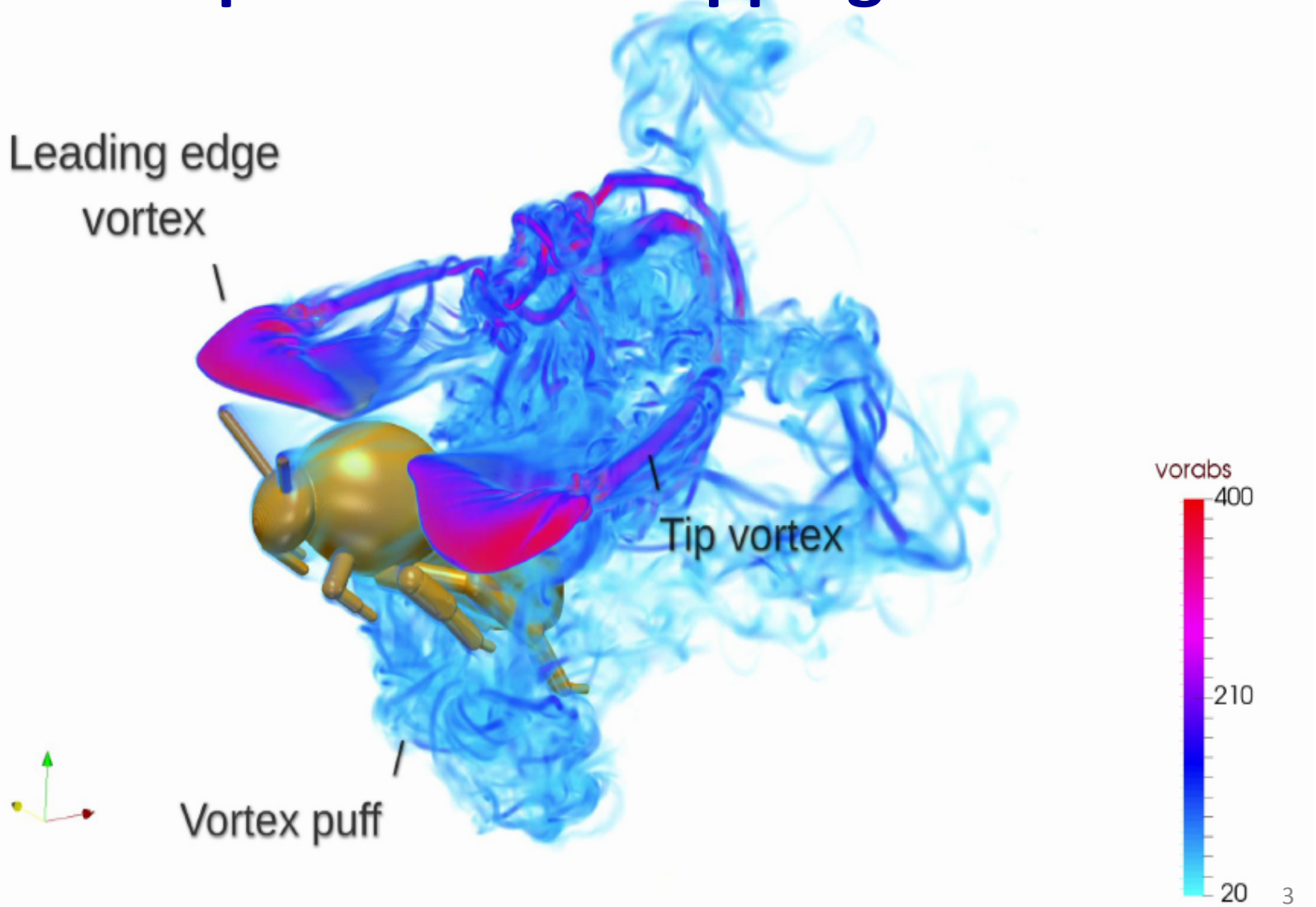
Wavelets and Applications

November 10-11, 2021, São José dos Campos, SP, Brazil

Outline

- Motivation
- Governing equations and numerical method
- Multiresolution analysis and grid adaptation
- Parallel implementation
- Numerical results and performance
- Bumblebee in the wake of a fractal tree
- Conclusions

Computations of flapping insects



Ref.: T. Engels, D. Kolomenskiy, K. Schneider, F.O. Lehmann and J. Sesterhenn. *Phys. Rev. Lett.*, 116, 028103, 2016.

Motivation

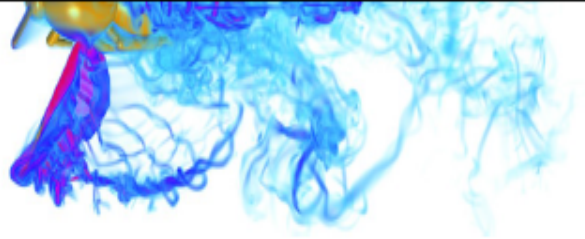
- Computing multiscale flows in complex geometries major challenge for computational fluid dynamics, especially in the turbulent flow regime

- Wavelets and related multiresolution analysis techniques provide likewise a mathematical framework and yield reliable error estimators, coupled with high computational efficiency; thus they are well suited for developing adaptive solvers with error control

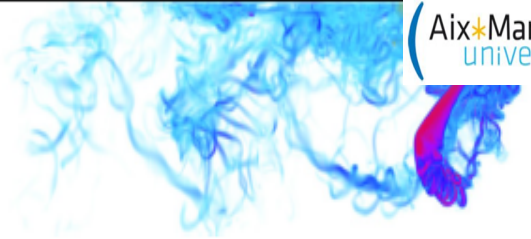
- boundary conditions for complex geometries, in particular at solid walls, the family of immersed boundary methods (IBM)

- Artificial compressibility method large but finite speed of sound and avoids solving elliptic equations

Ref.: T. Engels, K. Schneider, J. Reiss and M. Farge. A wavelet-adaptive method for multiscale simulation of turbulent flows in flying insects.
Commun. Comput. Phys., 30(4), 1118-1149, 2021. arXiv:1912:05371



Basic Idea



Aix*Marseille université

Observation: High resolution is not required everywhere at all times.

Idea: Refine grid where necessary.

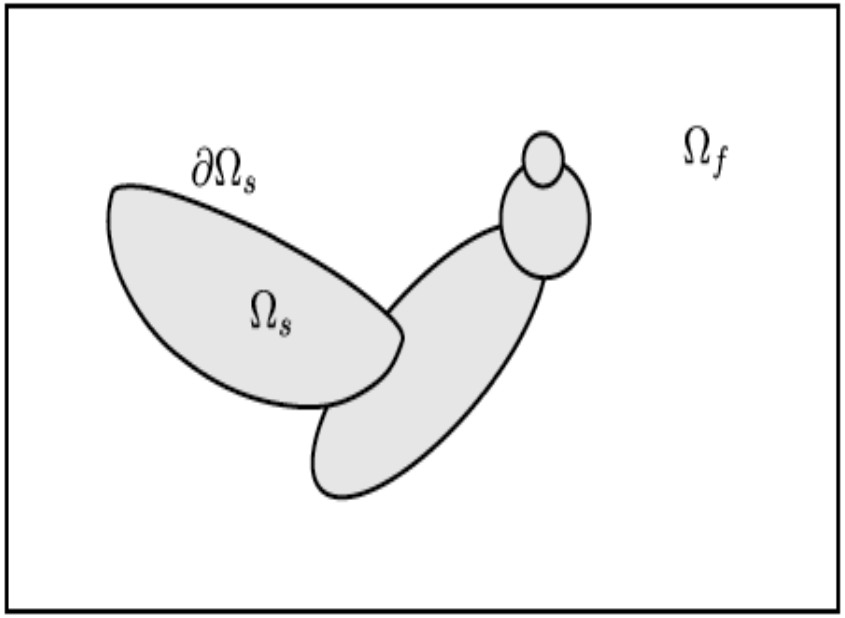
Question: How to know where refinement is required?

- Replace Fourier discretization by finite differences and allow local refinement / coarsening of the grid
- Use wavelets as local regularity estimators instead of the more heuristic criteria for adaptive mesh refinement (vorticity, strains, etc)
- We point out differences to other available adaptive methods

Physical Model: Governing Eqn.

- For simulations of insect flight, the fluid can be approximated as **incompressible** (here, density is normalized and s is a general coordinate)

$$
\begin{aligned}
\partial_t \underline{u} + \underline{u} \cdot \nabla \underline{u} &= -\nabla p + \nu \nabla^2 \underline{u} \\
\nabla \cdot \underline{u} &= 0 \\
\underline{u}|_{\partial\Omega_s(s,t)}(s,t) &= \underline{u}_s(s,t) \\
\underline{u}(\underline{x}, t=0) &= \underline{u}_0(\underline{x})
\end{aligned}
$$



Physical Model: Incompressibility

- For incompressible flow, a **Poisson** equation usually has to be solved, which is very demanding on adaptive grids
- A way to overcome this difficulty is using the method of **artificial compressibility** (ACM) [1,2] (some similarities with the Lattice-Boltzmann Method)

$$
\begin{aligned}
\partial_t \underline{u} &= -\underline{u} \cdot \nabla \underline{u} - \nabla p + \nu \nabla^2 \underline{u} \\
\partial_t p &= -C_0^2 \nabla \cdot \underline{u} \\
\underline{u}|_{\partial\Omega_s(s,t)}(s,t) &= \underline{u}_s(s,t) \\
\underline{u}(\underline{x}, t=0) &= \underline{u}_0(\underline{x}) \\
p(\underline{x}, t=0) &= p_0(\underline{x})
\end{aligned}
$$

- Ohwada [2] showed that this method is **second order**, (but their test cases are a bit weak)

$$
\|\underline{u}_{\mathrm{ACM}} - \underline{u}_{\mathrm{INC}}\|_2 = \mathcal{O}(C_0^{-2}) \quad \text{as } C_0 \to \infty
$$

[1] Chorin. A numerical method for solving incompressible viscous flow problems, J. Comput. Phys., **1967**.
[2] Ohwada et al., Artificial compressibility method revisited: Asymptotic numerical method for incompressible Navier–Stokes equations, J. Comp. Phys. **2010**

Physical Model: Divergence transport

- Often, ACM was used with subiterations (pseudo-time), e.g. [4]
- ACM is a transport of divergence: requires some outflow or damping for removal. Ohwada proposes dashpot-damping, works for periodic flows

$$\begin{aligned} \partial_t \underline{u} &= -\underline{u} \cdot \nabla \underline{u} - \nabla p + \nu \nabla^2 \underline{u} \\ \partial_t p &= -C_0^2 \nabla \cdot \underline{u} - C_\gamma p \end{aligned}$$

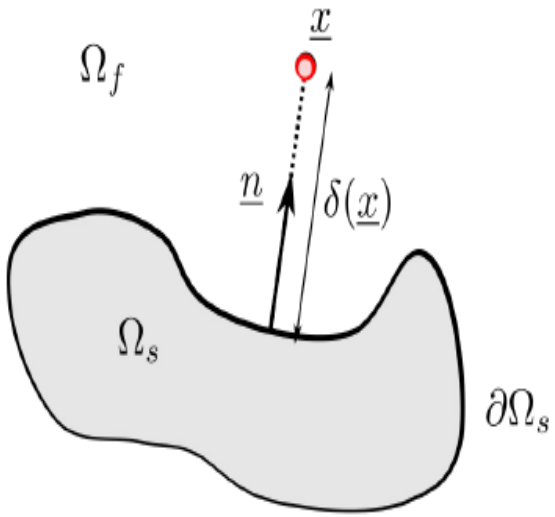
- We prefer non-reflecting outflow boundary conditions in most cases (exterior flows)
- Simple, explicit, 4th order finite difference discretization [3]
- Boundary conditions enforced with penalization method.

[3] Tam, Webb, Dispersion-Relation-Preserving Finite Difference Schemes for Computational Acoustics, J. Comp. Phys. **1993**
[4] H. Liu, Integrated modeling of insect flight: From morphology, kinematics to aerodynamics, J. Comp. Phys **2009**

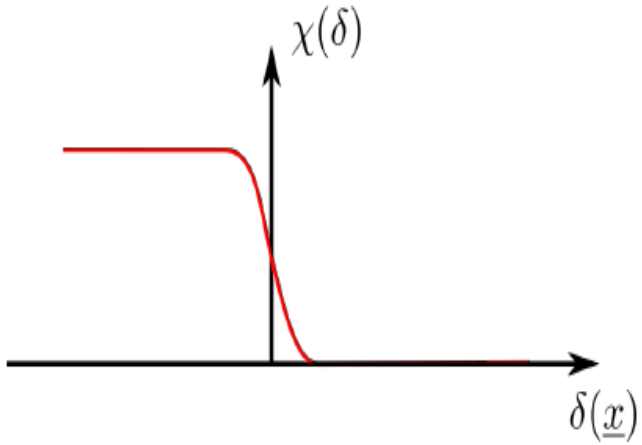
Physical Model: Penalization

- Penalization is mathematically justified and physically inspired

$$\partial_t \underline{u} = \ldots - \underbrace{\frac{\chi}{C_\eta}\left(\underline{u} - \underline{u}_s\right)}_{\text{penalization}}$$



$$\chi(\delta) = \begin{cases} 0 & \text{in } \Omega_f \\ 0 < \chi < 1 & \text{near } \partial\Omega_s \\ 1 & \text{in } \Omega_s \end{cases}$$



Engels et al. FluSI: A novel parallel simulation tool for flapping insect flight using a Fourier method with volume penalization. SIAM, **2016**

Angot et al. A penalization method to take into account obstacles in incompressible viscous flows. Numer. Math., **1999**

Complete Physical Model: Parameters

- The complete physical model is as follows:

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

$$\partial_t p = -C_0^2\nabla\cdot\underline{u} - C_\gamma p - \frac{\chi_{\mathrm{sp}}}{C_{\mathrm{sp}}}(p-p_\infty)$$

- The parameters need to be fixed. Physical intuition gives ideas

$C_0 \gg |\underline{u}|$

$C_0 < a_0$

$C_\eta \to 0$

$C_\gamma = 0$ for exterior flows

$C_{\mathrm{sp}} = 0, C_\gamma = 1$ for periodic flows

- Choice of penalization parameter is solved problem: (error balancing)

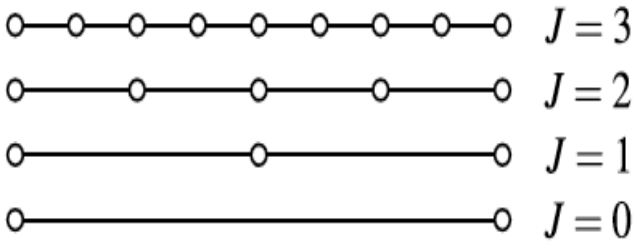
$$C_\eta = \left(K_\eta^2/\nu\right)\Delta x^2$$

How to introduce adaptivity ?

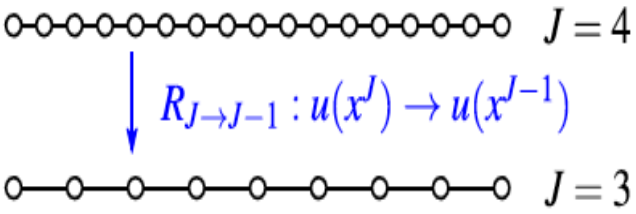
Multiresolution

- We can base the decision which grid points are important on Hartens point-value multiresolution [1]. We show later how it relates to biorthogonal wavelets
- Starting from a **hierarchy of nested grids** with odd number of points, here 1D:

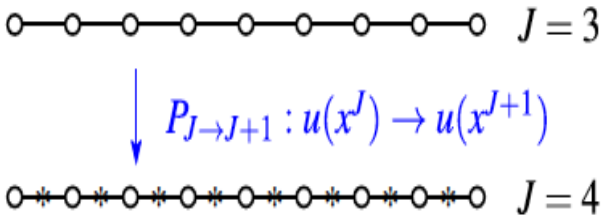
$$X^J = \left\{x_i^J\right\}_{i=0}^{2^J} = \left\{2^{-J} i\right\}_{i=0}^{2^J}$$



- Define **restriction** operator:



Define **prediction** operator:

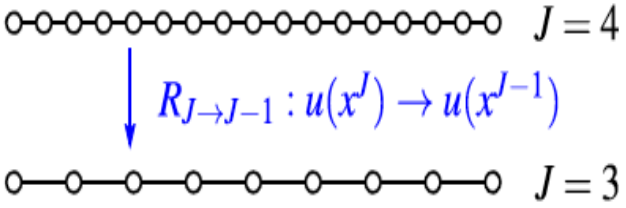


- Restriction: low pass or simple decimation, Prediction: interpolation with some order

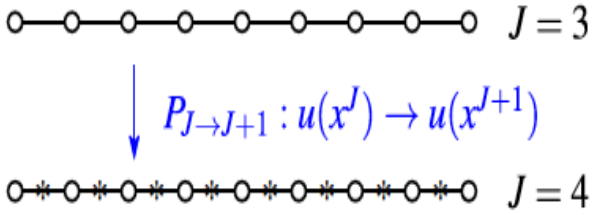
[1] Harten, Discrete multi-resolution analysis and generalized wavelets, Appl. Numer. Anal. **1993**
[2] Deslauriers & Dubuc, Symmetric iterative interpolation processes, Constr. Approx. **1989**

Multiresolution

- Define restriction operator:

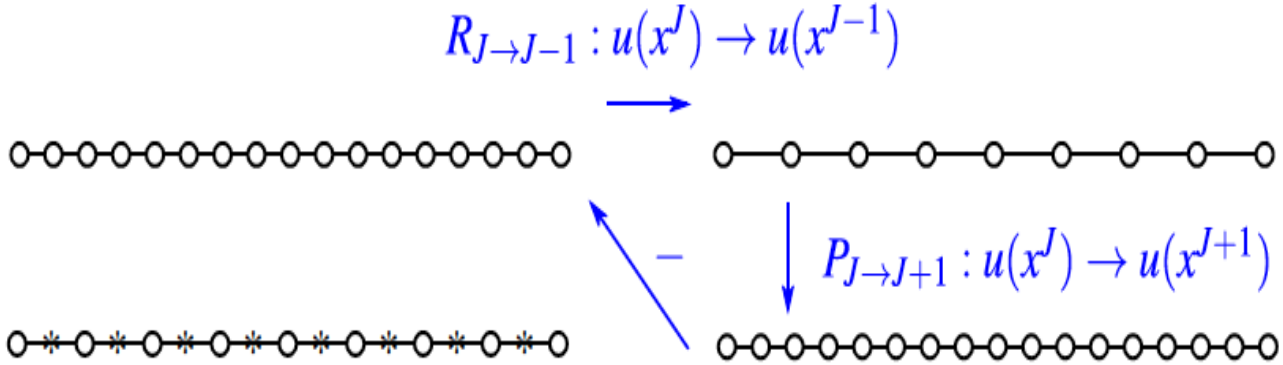


Define prediction operator:



- Computation of **Detail** coefficients:

$$d_{2i-1}^{J-1} = u_i^J - P_{J-1\to J} R_{J\to J-1} u_i^J$$



Harten, Discrete multi-resolution analysis and generalized wavelets, Appl. Numer. Anal. **1993**
Deslauriers & Dubuc, Symmetric iterative interpolation processes, Constr. Approx. **1989**
Domingues et al. Adaptive Multiresolution methods. ESAIM Proceedings **2011**

Multiresolution

- Each point can be associated with a detail coefficient, and we can decide:

$|d(x)| \geq C_\varepsilon \rightarrow$ keep point

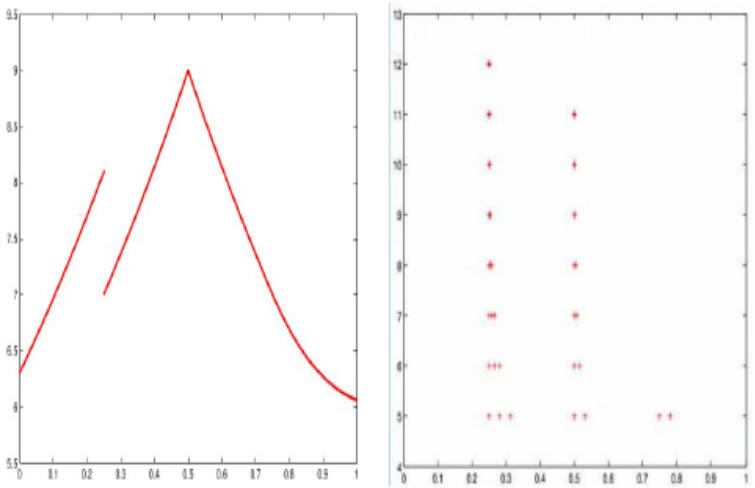
$|d(x)| < C_\varepsilon \rightarrow$ discard point

Thresholding of the wavelet coefficients

- Then the reconstruction error is bounded by:

$$\|u(x) - u^\varepsilon(x)\|_\infty \leq KC_\varepsilon$$

- Decay of details is related to local regularity:



Domingues et al. Adaptive Multiresolution methods. ESAIM Proceedings **2011**

FIGURE 6. The function $f(x)$ (left) and the position of the significant wavelet coefficients $|d_{j,k}| \leq 5 \times 10^{-4}$ (right).

Multiresolution and Wavelets

- Biorthogonal wavelets decompose a function in scaling + wavelet coefficients

$$f(x) = \sum_{i\in\mathbb{Z}} \left\langle f, \widetilde{\phi_i^0}(x) \right\rangle \phi_i^0(x) + \sum_{J=0}^{\infty} \sum_{i\in\mathbb{Z}} \left\langle f, \widetilde{\psi_i^J}(x) \right\rangle \psi_i^J(x)$$

$$\left\langle \phi(x-i)\widetilde{\phi}(x-j) \right\rangle = \delta_{i,j} \qquad \left\langle \psi(x-i)\widetilde{\psi}(x-j) \right\rangle = \delta_{i,j}$$

- Wavelets ψ and scaling functions ϕ (and dual functions $\widetilde{\psi}$, $\widetilde{\phi}$) are defined with filters (high- and low pass). These can be obtained by re-arranging Hartens MR.

Schneider, Vasilyev. Wavelet Methods in Computational Fluid Dynamics, Ann. Rev. Fluid Dyn. **2010**.

Lifted biorthogonal wavelets

To increase (or add) number of vanishing moments to the point value MRA we use lifting:

More precisely we determine lifting coefficients s_n such that

$$\psi(x) = \psi^{\text{old}}(x) - \sum_n s_n \phi^{\text{old}}(2x-n), \tag{3.6}$$

$$\tilde{\phi}(x) = \sum_n \tilde{h}_n \tilde{\phi}^{\text{old}}(2x-n) + \sum_n s_n \psi^{\text{old}}(x-n), \tag{3.7}$$

requiring that the new ψ has M vanishing moments, *i.e.*, $\int x^p \psi(x)dx = 0$ for $p = 0, \cdots, M-1$. In the linear interpolation case, *i.e.*, CDF 2/0, we have $s_0 = s_1 = -1/4$ and for the resulting lifted filters we get $\{g_n;\ n = -3, \cdots, 1\} = \{1/8, -1/4, 3/4, -1/4, -1/8\}$ and $\{\tilde{h}_n;\ n = -2, \cdots, 2\} = \{1/8, -1/4, -3/4, -1/4, 1/8\}$, yielding thus CDF 2/2 wavelets, which have two vanishing moments. Applying lifting to the cubic interpolatory wavelets CDF 4/0 yields CDF 4/4.

Ref.: W. Sweldens. The lifting scheme: A construction of second generation wavelets. SIAM J. Math. Anal. , 29(2):511–546, 1989.

CDF wavelets

Table 1: Filter coefficients for the CDF4/0 and CDF4/4 interpolating biorthogonal wavelets.

	CDF4/0				CDF4/4			
i	$\widetilde{h}$	$\widetilde{g}$	h	g	$\widetilde{h}$	$\widetilde{g}$	h	g
-7								−1/256
-6					−1/256			0
-5					0			9/128
-4					9/128			1/16
-3			−1/16		−1/16		−1/16	−63/256
-2		1/16	0		−63/256	1/16	0	−9/16
-1		0	9/16		9/16	0	9/16	87/64
0	1	−9/16	1		87/64	−9/16	1	−9/16
1		1	9/16	1	9/16	1	9/16	−63/256
2		−9/16	0		−63/256	−9/16	0	1/16
3		0	−1/16		−1/16	0	−1/16	9/128
4		1/16			9/128	1/16		0
5					0			−1/256
6					−1/256			

CDF wavelets

CDF 4/0

CDF 4/4

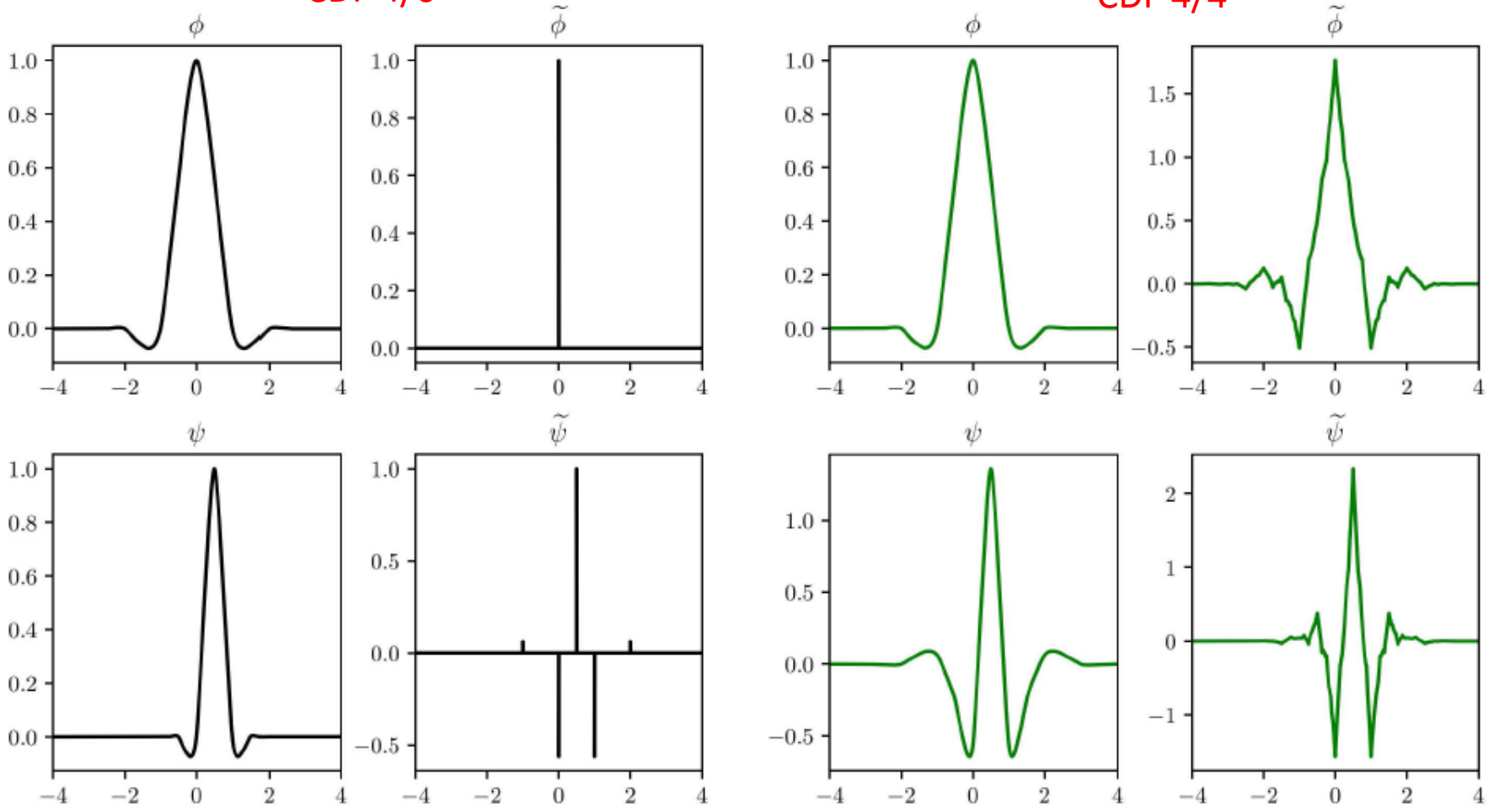


Figure 2: Scaling functions ϕ, wavelets ψ and dual functions $\widetilde{\phi}$, $\widetilde{\psi}$ for the CDF 4/0 (left, black) and CDF 4/4 wavelets (right, green).

Approximation of first order derivatives

In the present work, we use the optimized fourth order scheme proposed by Tam and Webb [6] for first derivatives. It is built on the idea of combining a fourth- and a sixth order approximation using standard finite differences, yielding

$$(\partial_x u)_i \approx \Delta x^{-1} \left(\sum_j \left(\gamma a_j^{(4\text{th})} + (1-\gamma) a_j^{(6\text{th})} \right) u_{i+j} \right),$$

where $a_j^{(4\text{th})}$ and $a_j^{(6\text{th})}$ are classical, central finite differences. The combination provides a degree of freedom γ, which is then used to optimize the modified wavenumber. The resulting stencil is

$$a_j^{(\text{TW})} = -a_{-j}^{(\text{TW})} = \{-0.02651995, 0.18941314, -0.79926643, 0\}, \quad (-3 \le j \le 0).$$

Ref.: C. K.W. Tam and J. C. Webb. Dispersion-relation-preserving finite difference schemes for computational acoustics. J. Comput. Phys. , 107(2):262–281, 1993.

For second order derivatives, classical centered fourth order schemes are used.

Parallelization

Data Structure

- That sounds great, how come people still use non-adaptive grids?
  - Caching efficiency: RAM is 1D, CPU reads memory pages
  - Still, most algorithms are memory bound, even with regular datastructures (exception: Fourier spectral methods)
  - Efficiency of unstructured / sparse grids is typically low
- Early adaptive codes used dense data structures [1]
- Block based data structure as a compromise between CPU requirements and theoretical efficiency [2]
  - Separation into heavy and light data
  - Use tree-like indexing for block management (slow, but far less blocks than points) [3]
- In the following, we present the ideas on which our new code "WABBIT" (**W**avelet **A**daptive **B**lock-**B**ased Solver for **I**nteractions/Insects with **T**urbulence) [4]
- Open source and freely available on https://github.com/adaptive-cfd/WABBIT

[1] Holmström. Solving hyperbolic PDEs using interpolating wavelets SIAM J. Sci. Comp **1999**

[2] Domingues et al. Adaptive wavelet representation and differentiation on block-structured grids. Appl. Numer. Math. **2003**

[3] Gargantini. An effective way to represent quadtrees. Commun. ACM **1982**

[4] Engels et al. Wavelet-based adaptive simulations of multiscale flows around complex geometries. **arXiv:1912.05371**

Data Structure

Grid layout

Memory layout

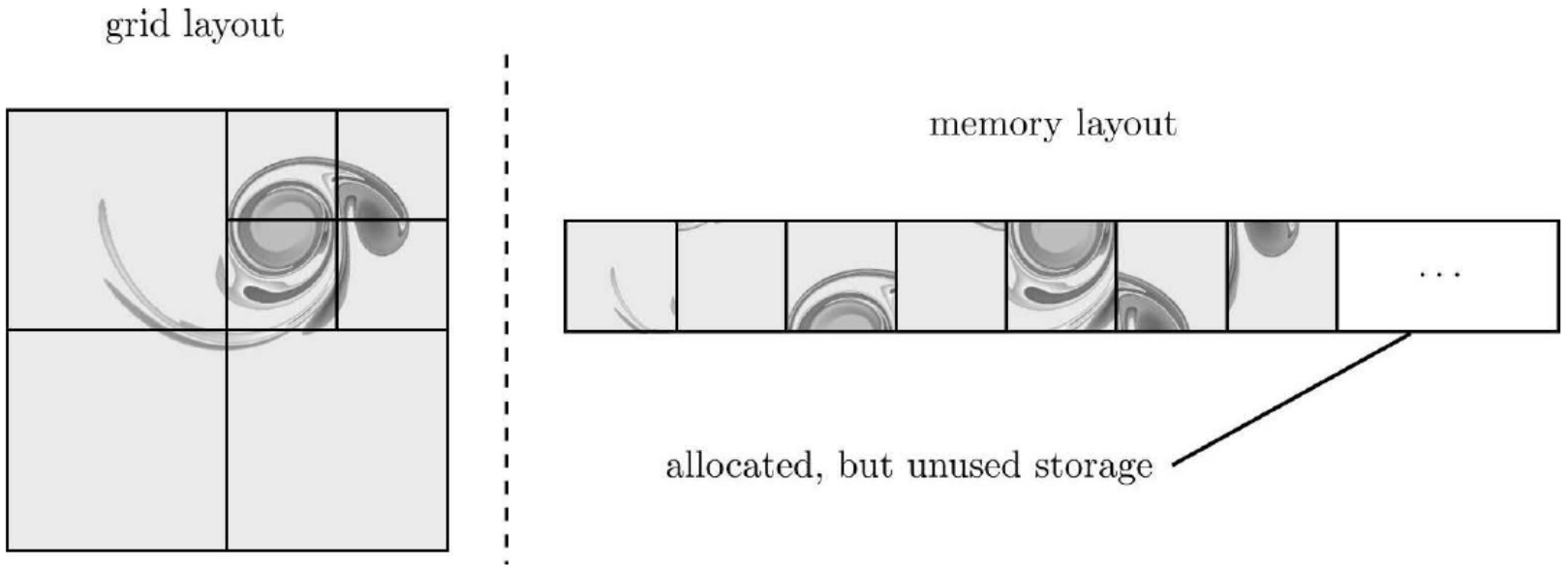
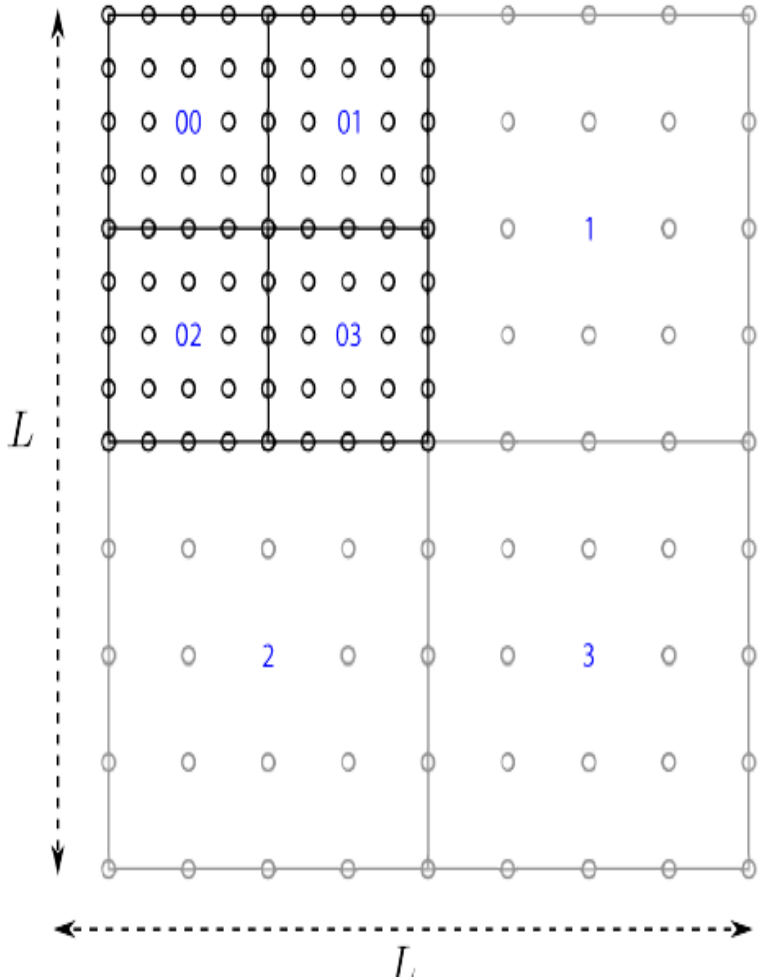


Figure 1: Data structure. Left: grid layout, composed of N_b=7 blocks of the same size but at different positions and levels. Right: memory layout, the grid is stored as simple, contiguous array. Memory allocation done once on startup, but only the portion required at time t is used.

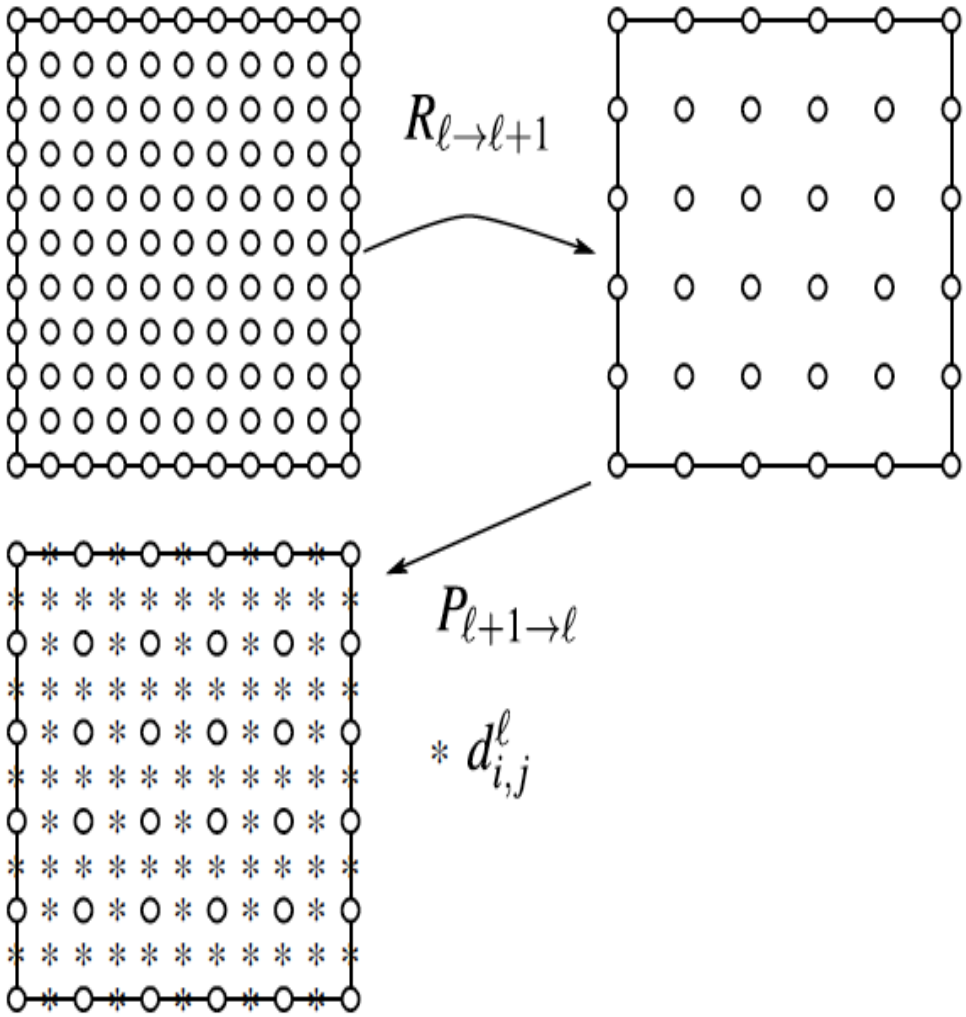
Example grid

- The grid is composed of blocks of size B_s^D
- Gradedness: no more than a factor of two in spacing between two blocks

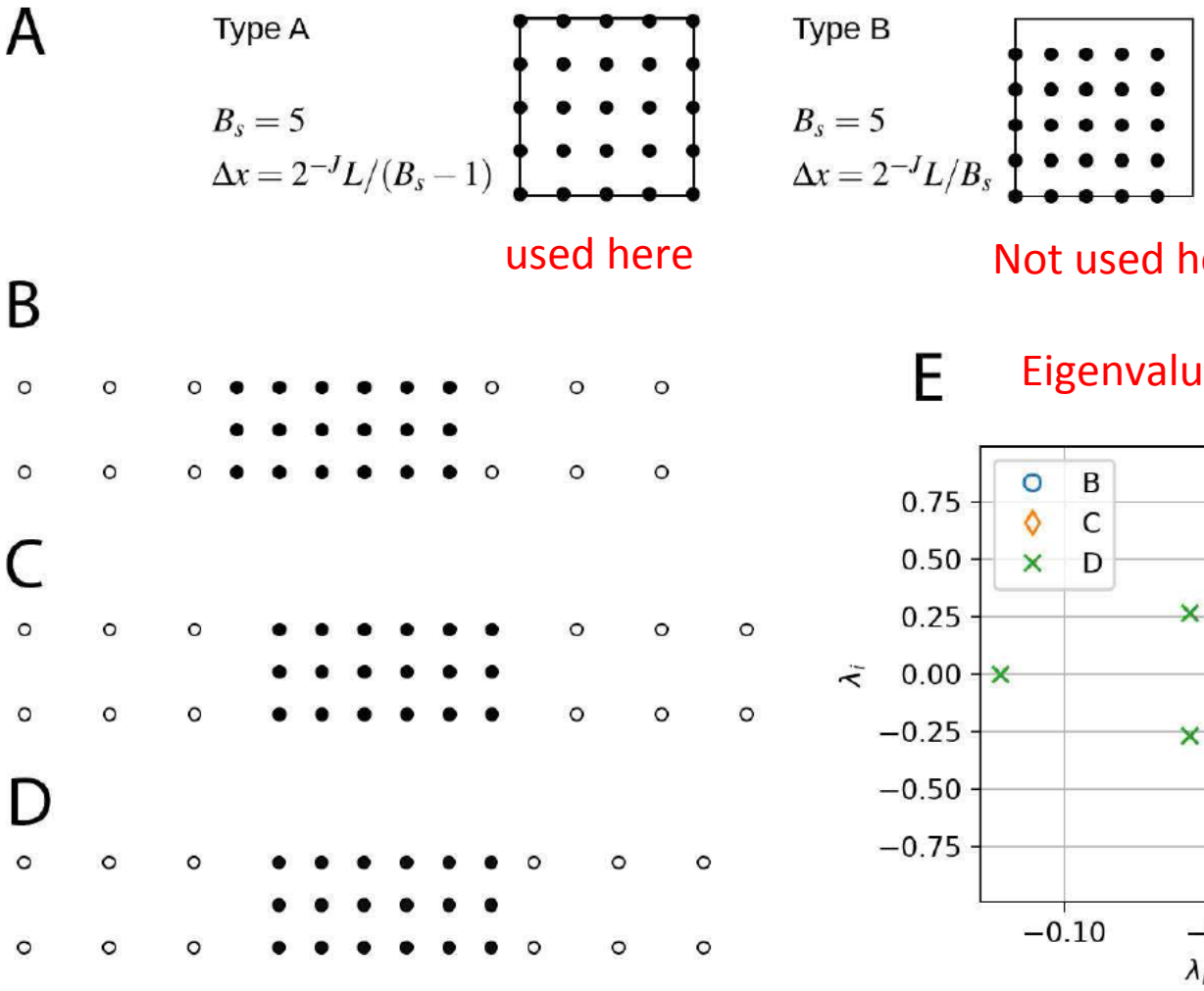


Computation of Details

- Detail coefficients are obtained by coarsening, then refining (interpolation)
- Their magnitude is related to the local regularity of the solution
- Largest detail coefficient determines state of the block



Design of the computational grid



Non zero real parts for case D, unstable

Figure 1: A: Two possible block definitions using the same block size B_s. Type A includes all of the block borders, which are then redundant with the neighboring block, while type B assigns each block border to one block. Present work uses type A grids exclusively. B-D: Different grid definitions for a coarse (open circles) and fine block (full circles). Ghost nodes are not shown. B: spacing between the blocks corresponds to fine block spacing on both sides. C: same as B, but using coarse block spacing. D: mixed interblock spacing E: eigenvalues of discrete first derivative (a Toeplitz matrix) for grids B-D in the complex plane.

Grid Prediction & Time Stepping Algorithm

The basic algorithm to advance the solution from t^n to t^{n+1} reads

1) Refinement stage. The entire grid is refined, to be sure that it is sufficient to contain the solution at the **new** time level (quadratic nonlinearities)
2) Evolution. First synchronize layer of ghost nodes on each block. Then solve the PDE using finite differences (4^{th} order) and advance in time (RK4)
3) Coarsening. Check the details and keep only those blocks where the details are significant, i.e. larger than C_ε
4) Load balancing. Distribute the blocks among the MPI processes

Liandrat, Tchamitchian, Resolution of the 1D regularized Burgers equation using a spatial wavelet approximation, NASA Contractor Rep. **1990**

Harten, Multiresolution algorithms for the numerical solution of hyperbolic conservation laws, Comm. Pure Appl. Math. **1995**

Schneider, Vasilyev, Wavelet methods in computational fluid dynamics, Ann. Rev. fluid mech. **2010**

Error balancing

- We have the following scaling laws for the error contributions:

$$\|u_{\mathrm{ACM}} - u\| \overset{\mathrm{def}}{=} e_{\mathrm{ACM}} = \mathcal{O}(C_0^{-2})$$

$$\left\|u_{\mathrm{ACM}}^{\Delta x} - u_{\mathrm{ACM}}\right\| \overset{\mathrm{def}}{=} e_{\mathrm{FDM}} = \mathcal{O}(\Delta x^4)$$

$$\left\|u_{\mathrm{ACM}}^{\Delta x,\varepsilon} - u_{\mathrm{ACM}}^{\Delta x}\right\| \overset{\mathrm{def}}{=} e_{\mathrm{MR}} \leq N_t C_\varepsilon$$

(Note the thresholding error is a conservative estimation)

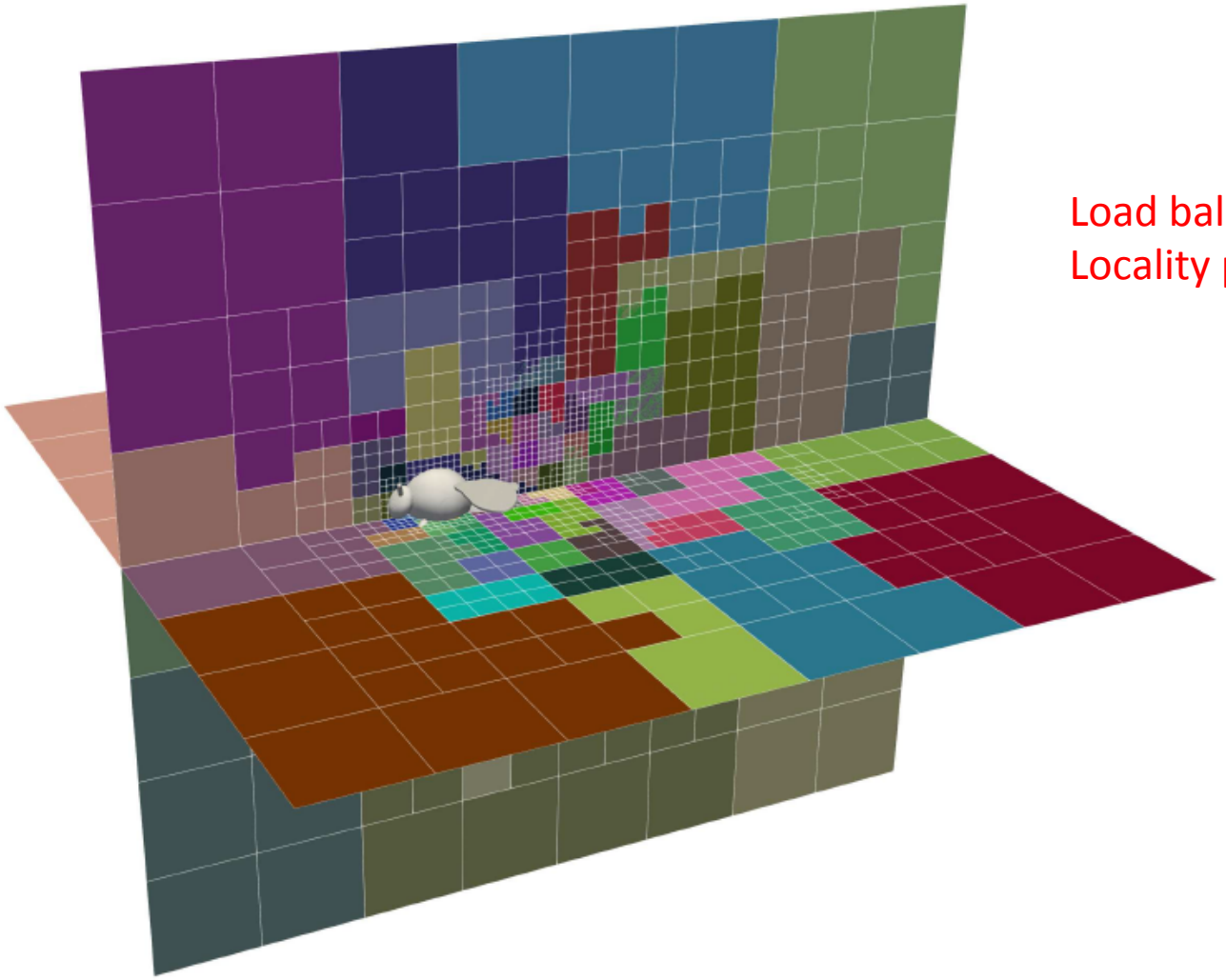
- To balance the errors, we are left with the following relations

$$C_0 \propto \Delta x^{-2}$$

$$C_{\varepsilon,\mathrm{opt}} \propto \Delta x^5$$

(Note $C_{\varepsilon,\mathrm{opt}} \propto \Delta x^4$ is a more optimistic but less rigorous estimate)

Parallelization aspects



Load balancing.
Locality preserving.

Figure 2: Parallel load balancing. Shown is a snapshot of the (fine resolution) bumblebee simulations presented in Section 5.3. In the two planes, the block borders are shown along with the `MPI` rank. Here, $N_{\mathrm{CPU}} = 960$, and each `CPU` is assigned a random color. The space-filling curve results in a locality-preserving distributions of blocks.

Numerical results

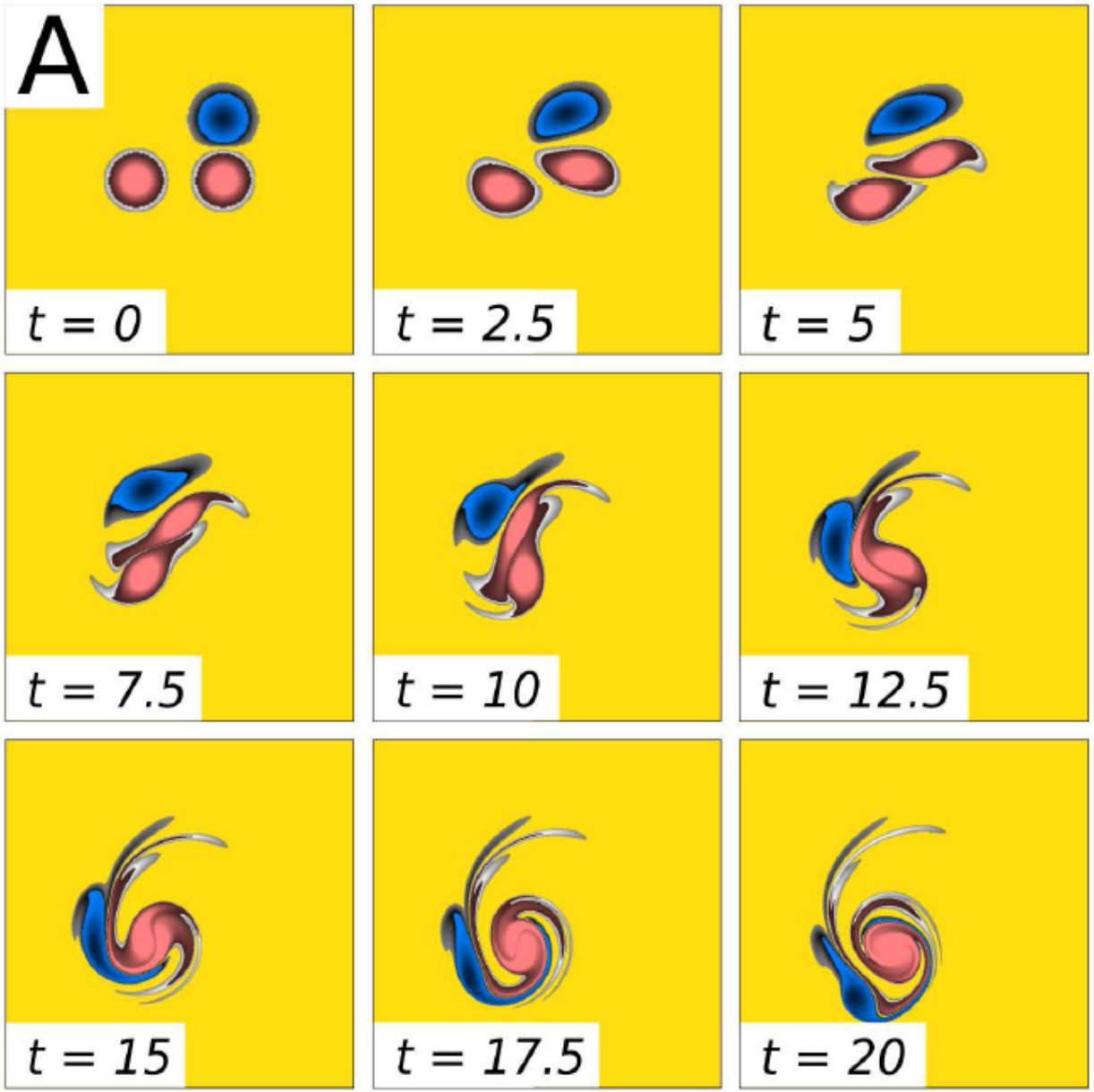
Three vortices (2D)

Four types of simulations:

1. Incompressible Navier–Stokes (INC) solved with a Fourier pseudospectral method [33], which is the quasi-exact reference solution;
2. ACM solved with the same Fourier pseudospectral method, as quasi-exact solution of the artificial compressibility equations;
3. ACM solved with the fourth-order finite difference method on equidistant grids, with and without filtering at the finest scale;
4. ACM solved with the fourth-order finite difference method on dynamically adaptive grids using multiresolution analysis with CDF 4/0 and CDF 4/4 wavelets.

Three vortices (2D)

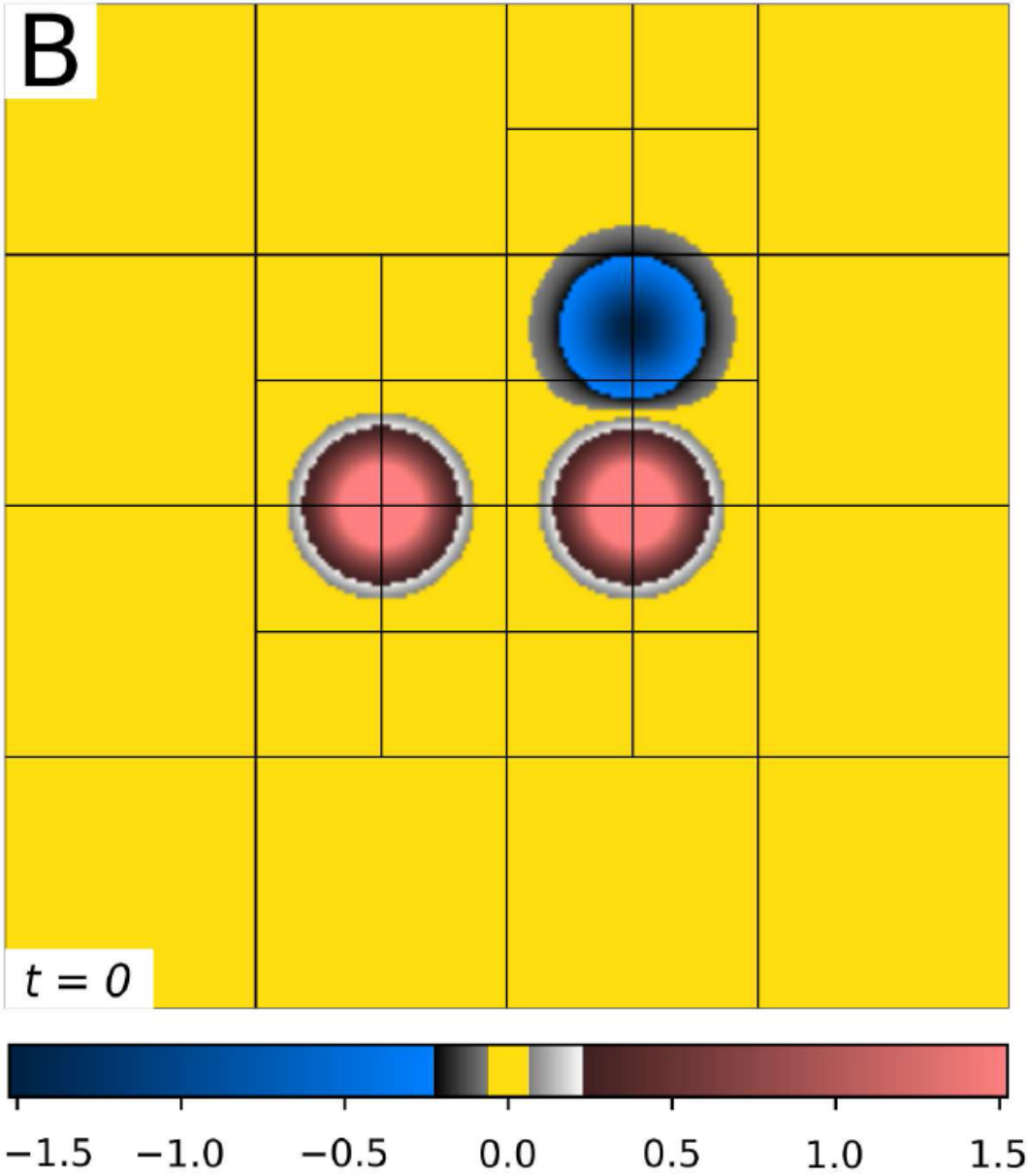
Time evolution of The vorticity field.



Three vortices (2D)

Initial condition.

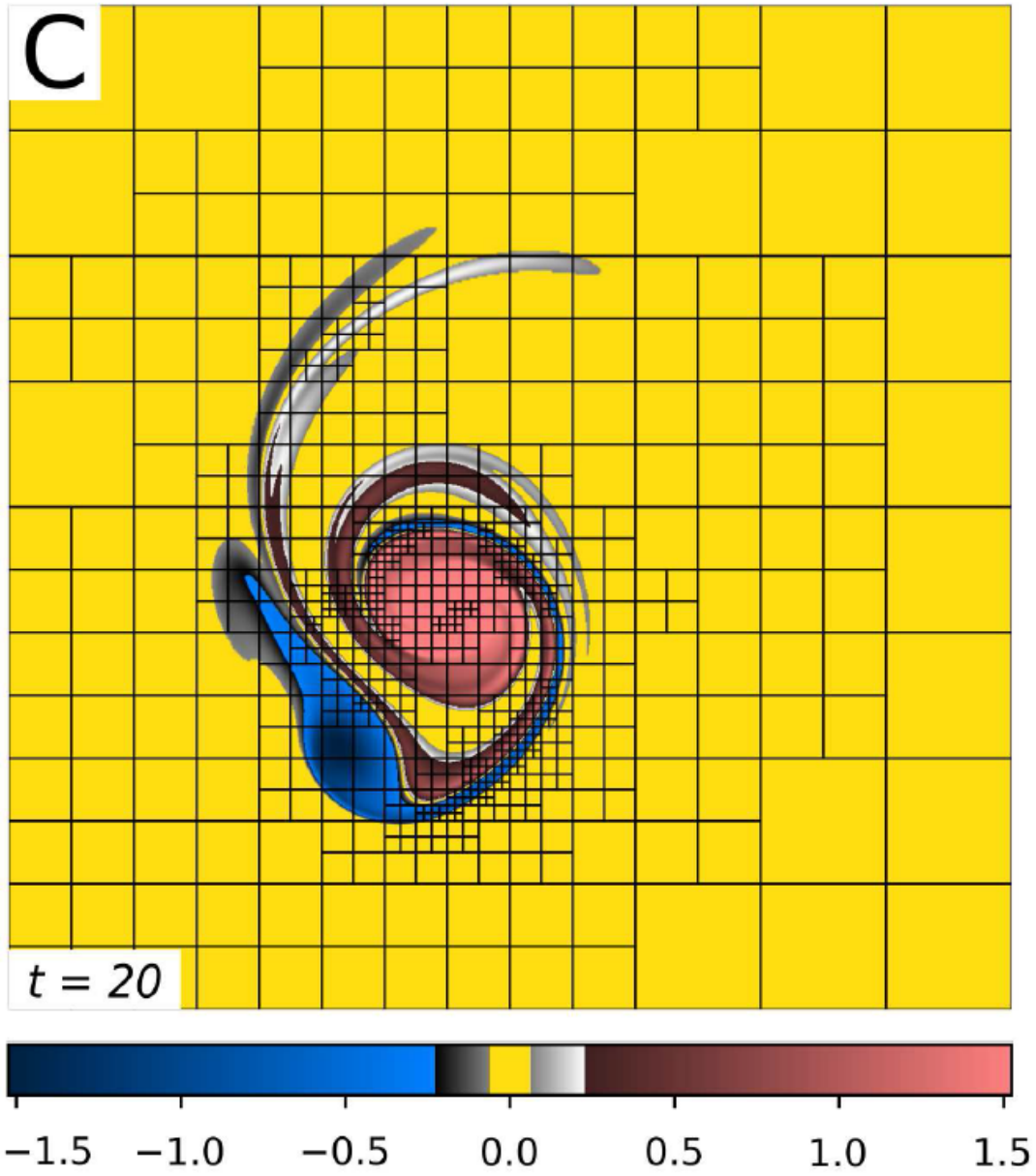
Vorticity field and Adaptive block grid



Three vortices (2D)

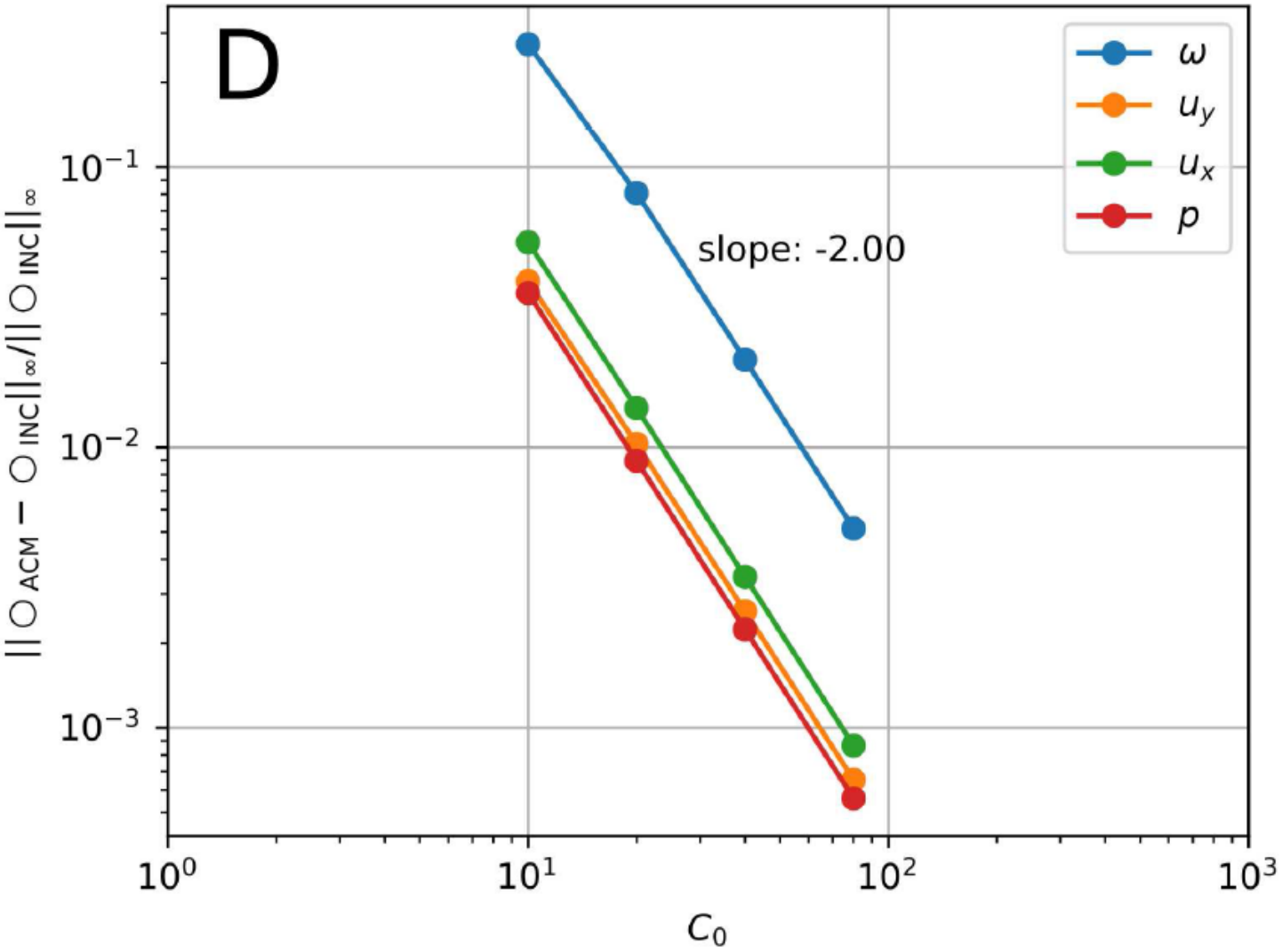
Final time used for error Computation.

Vorticity field and Adaptive block grid



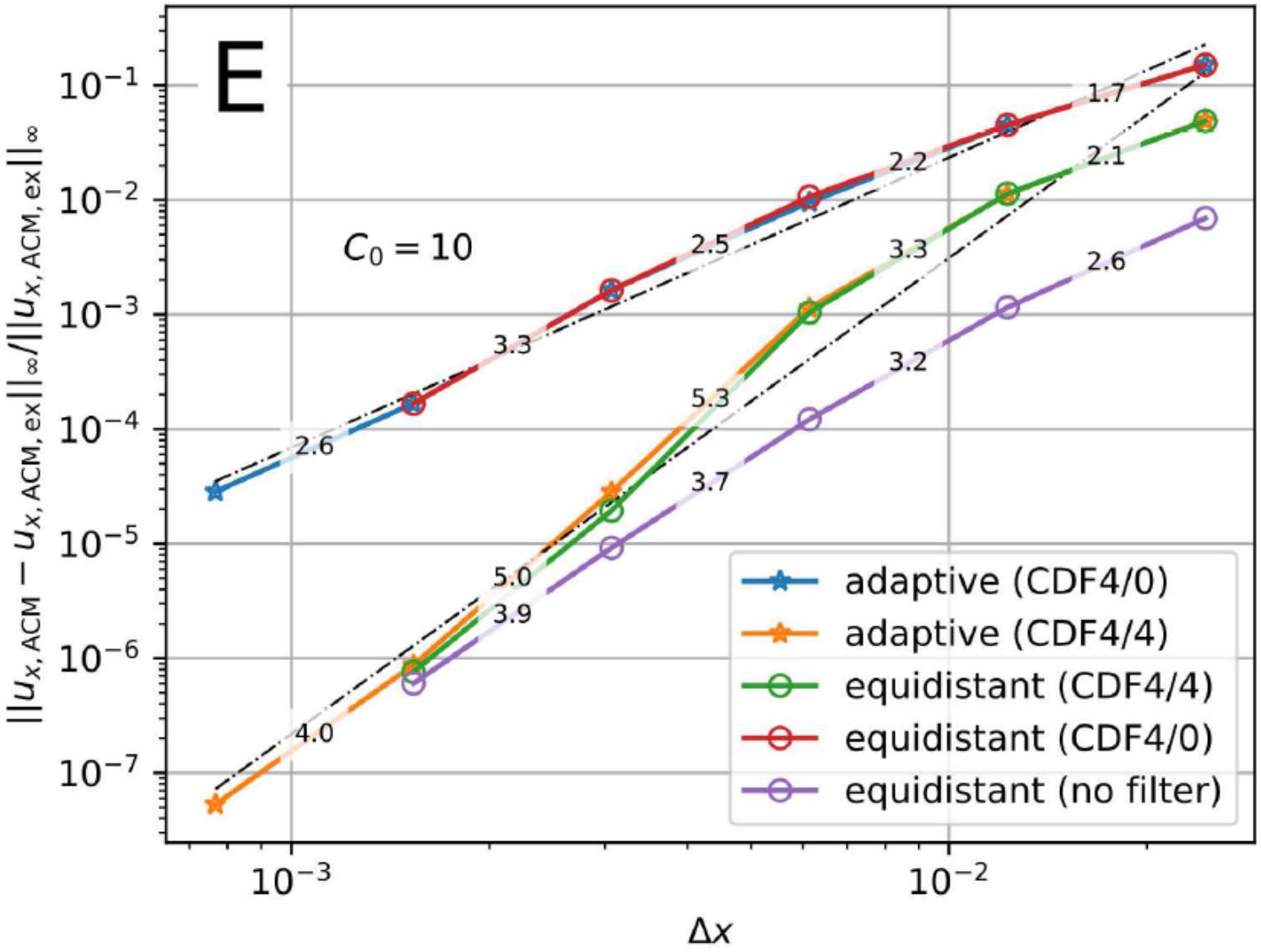
Three vortices (2D)

Decay of the compressibility error:



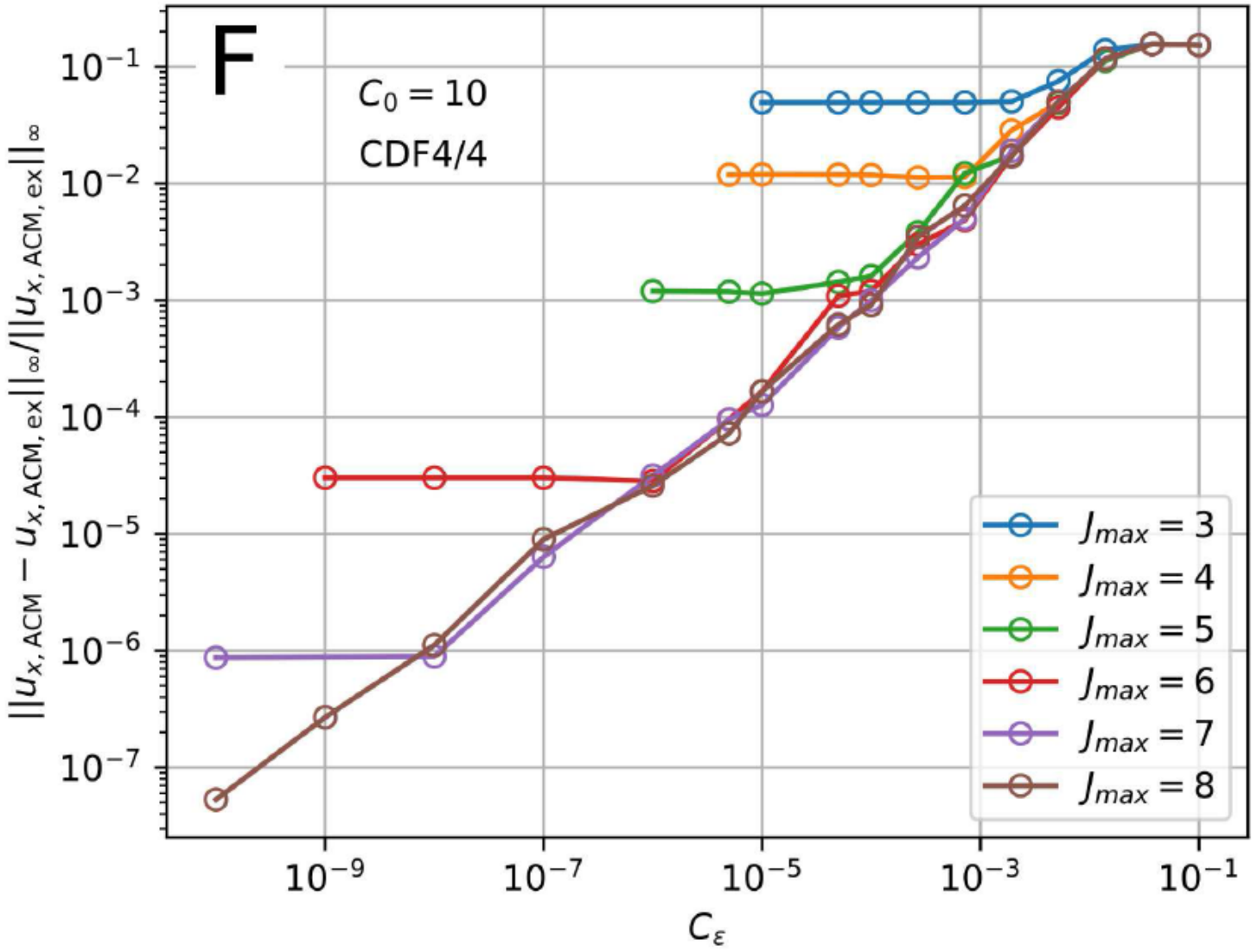
Three vortices (2D)

Decay of the error as a function of the grid size:



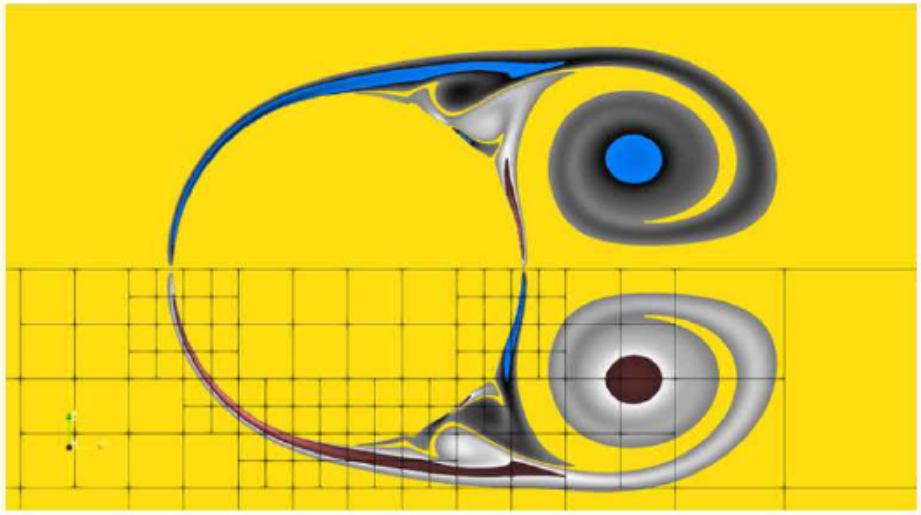
Three vortices (2D)

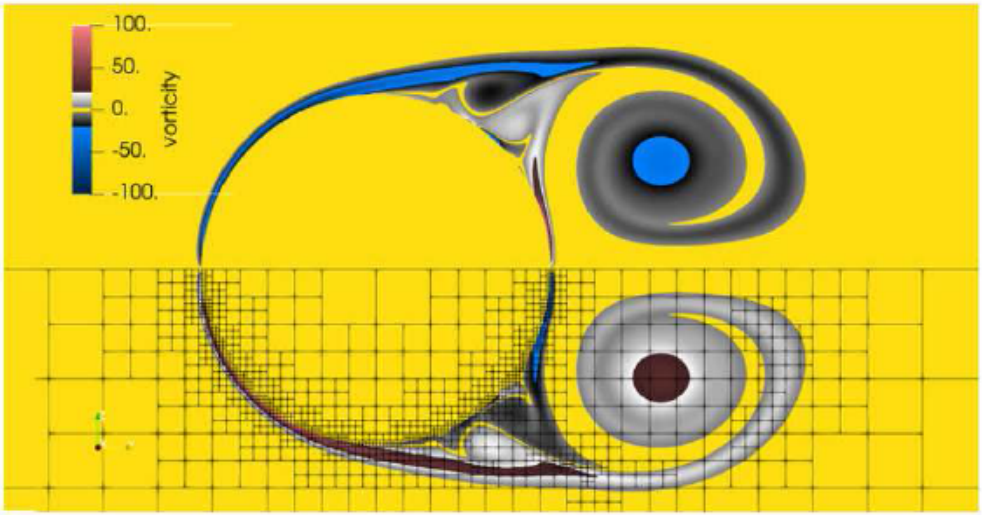
Decay of the error as a function of the thresholding parameter:



Impulsively started cylinder (2D)

Vorticity field with block-based grid superimposed in the lower half.





Level 1 computation (coarse).

Level 4 computation (fine).

Impulsively started cylinder (2D)

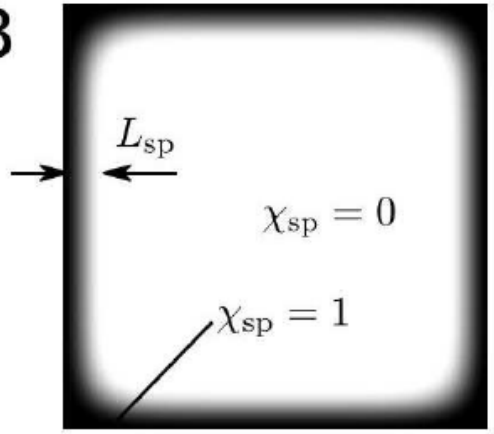
Computational parameters:

A

Parameter	level 1	level 2	level 3	level 4
$J_{\max}$	12	13	14	15
C_ε	10^{-3}	$2.50 \cdot 10^{-4}$	$6.25 \cdot 10^{-5}$	$1.56 \cdot 10^{-5}$
C_0	12.5	25	50	100
C_η	$1.01 \cdot 10^{-4}$	$2.51 \cdot 10^{-5}$	$6.29 \cdot 10^{-6}$	$1.57 \cdot 10^{-6}$
C_{sp}	$6.2 \cdot 10^{-2}$	$3.1 \cdot 10^{-2}$	$1.5 \cdot 10^{-2}$	$7.7 \cdot 10^{-3}$

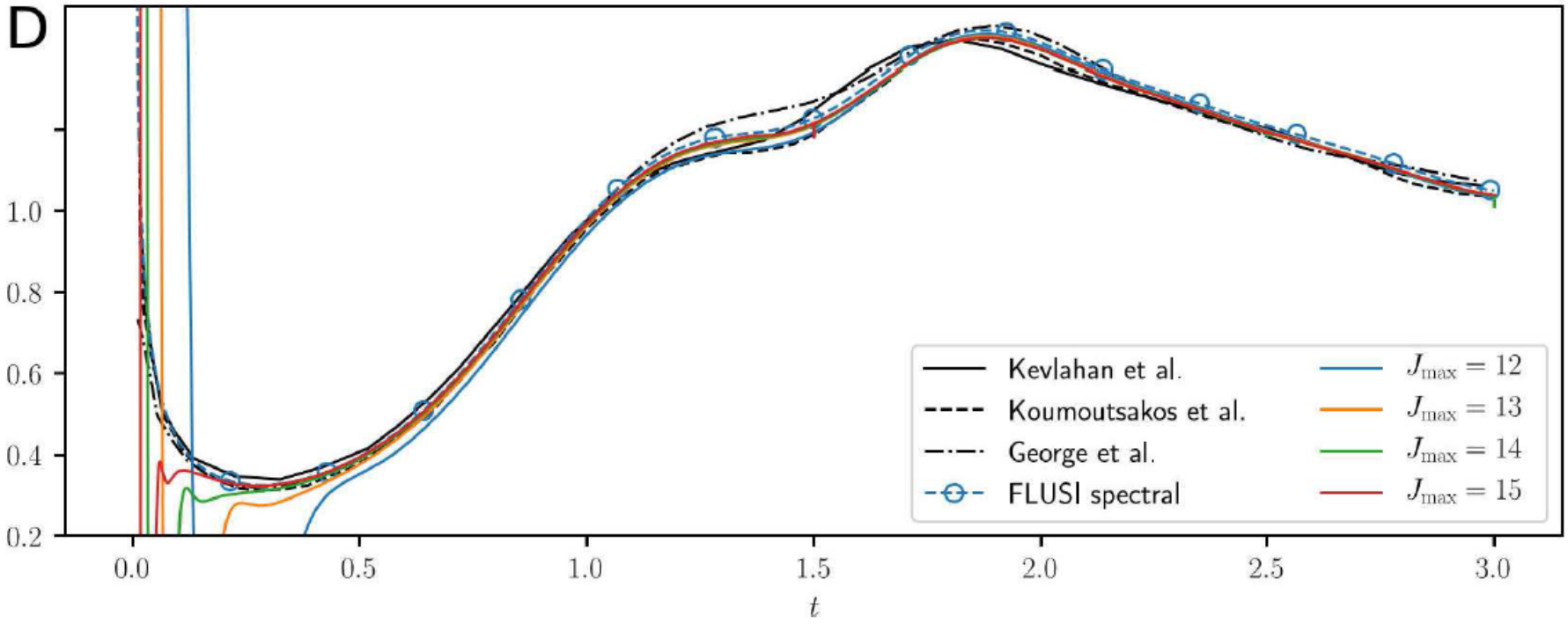
Mask function for the sponge.

B



Impulsively started cylinder (2D)

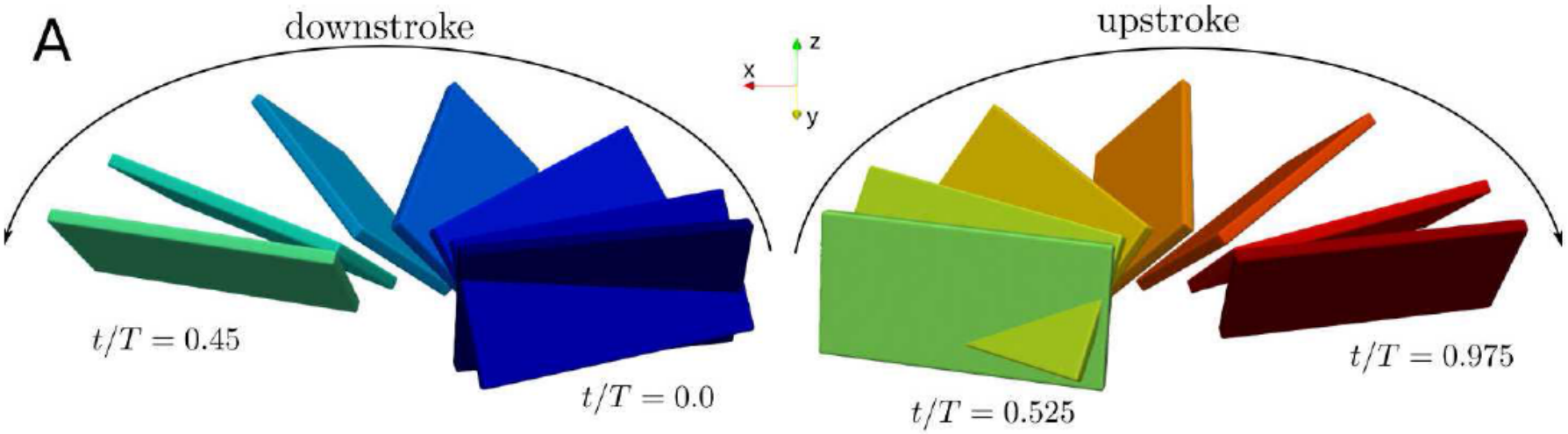
Drag coefficient as a function of time, compared with results from the literature.



Our results are obtained with CDF 4/0 wavelets.
Reasonable agreement after t=0.5.
ACM not well suited for impulsive start.

Flapping wing (3D)

Suzuki's test case. A: visualization of the wing beat.



The rigid wing moves in a horizontal stroke plane with varying angle of attach.
The motion is symmetrical in down and upstroke.

Ref.: K. Suzuki, K. Minami, and T. Inamuro. Lift and thrust generation by a butterfly-like Flapping wing-body model: Immersed boundary-lattice Boltzmann simulations. J. Fluid Mech. , 767:659–695, 2015.

Flapping wing (3D)

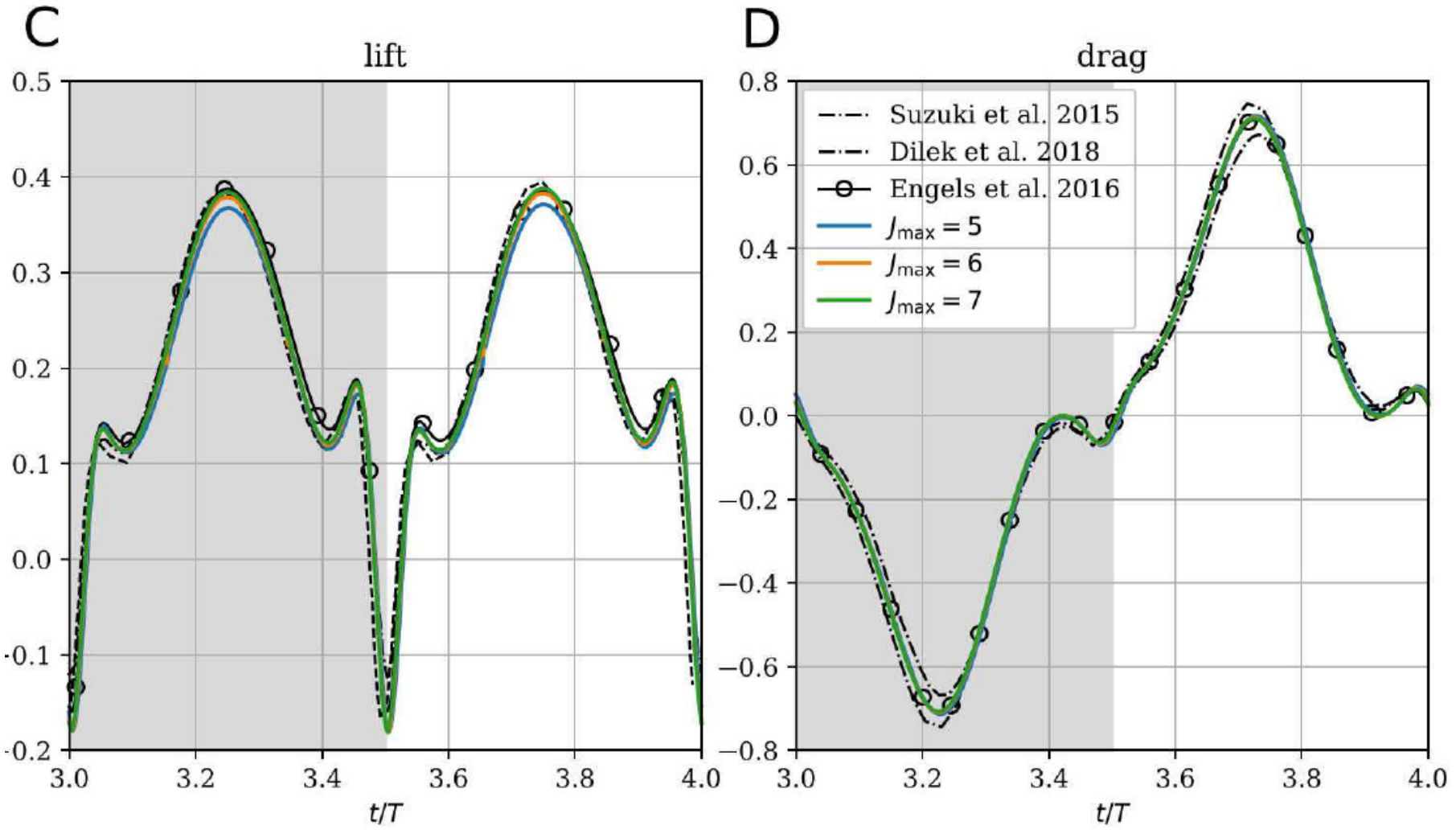
Computational parameters used in the simulations:

B

	$J_{\max}$	$h_{\text{wing}}/\Delta x$	C_ε	C_0	C_η	C_{sp}
coarse	5	4.9	$1.6 \cdot 10^{-2}$	25	$2.64 \cdot 10^{-4}$	$1.0 \cdot 10^{-3}$
medium	6	9.8	$4 \cdot 10^{-3}$	35.36	$6.61 \cdot 10^{-5}$	$7.1 \cdot 10^{-4}$
fine	7	19.6	10^{-3}	50	$1.65 \cdot 10^{-5}$	$5 \cdot 10^{-4}$

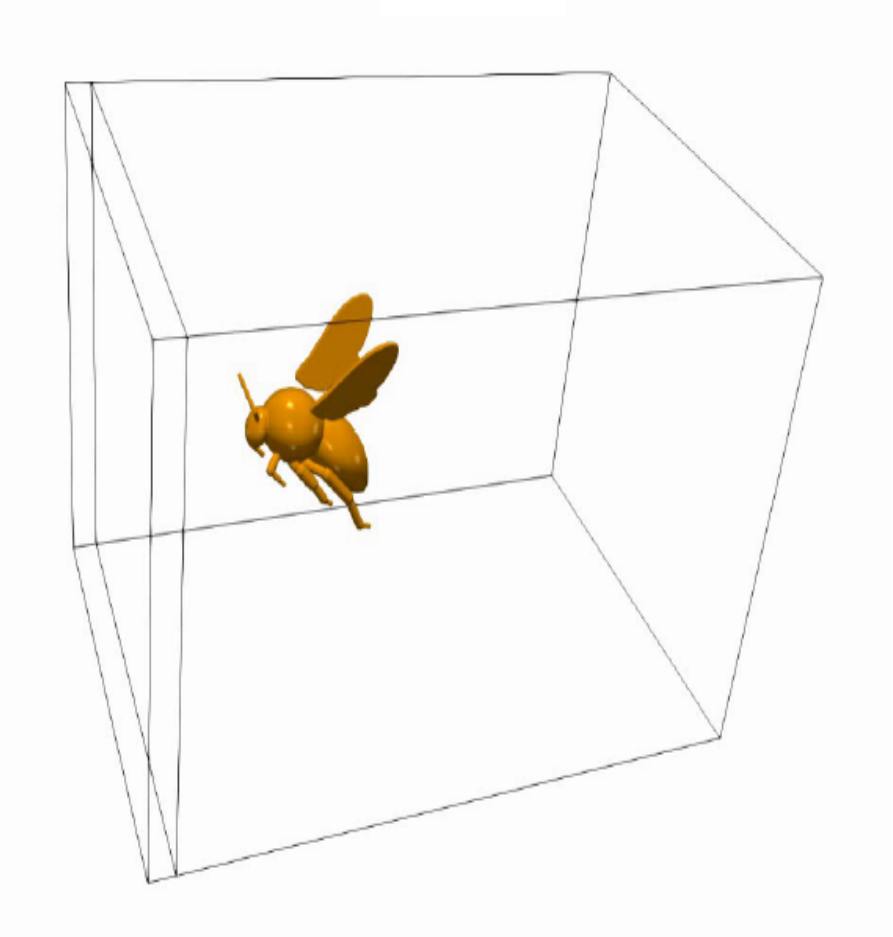
Flapping wing (3D)

Time evolution of lift and drag and comparison with results from the literature.



Bumblebee

Bumblebee model



$$R = 13.2\,\mathrm{mm}$$

$$f = 152\,\mathrm{Hz}$$

$$u_\infty = 2.5\,\mathrm{m/s}$$

$$m = 175\,\mathrm{mg}$$

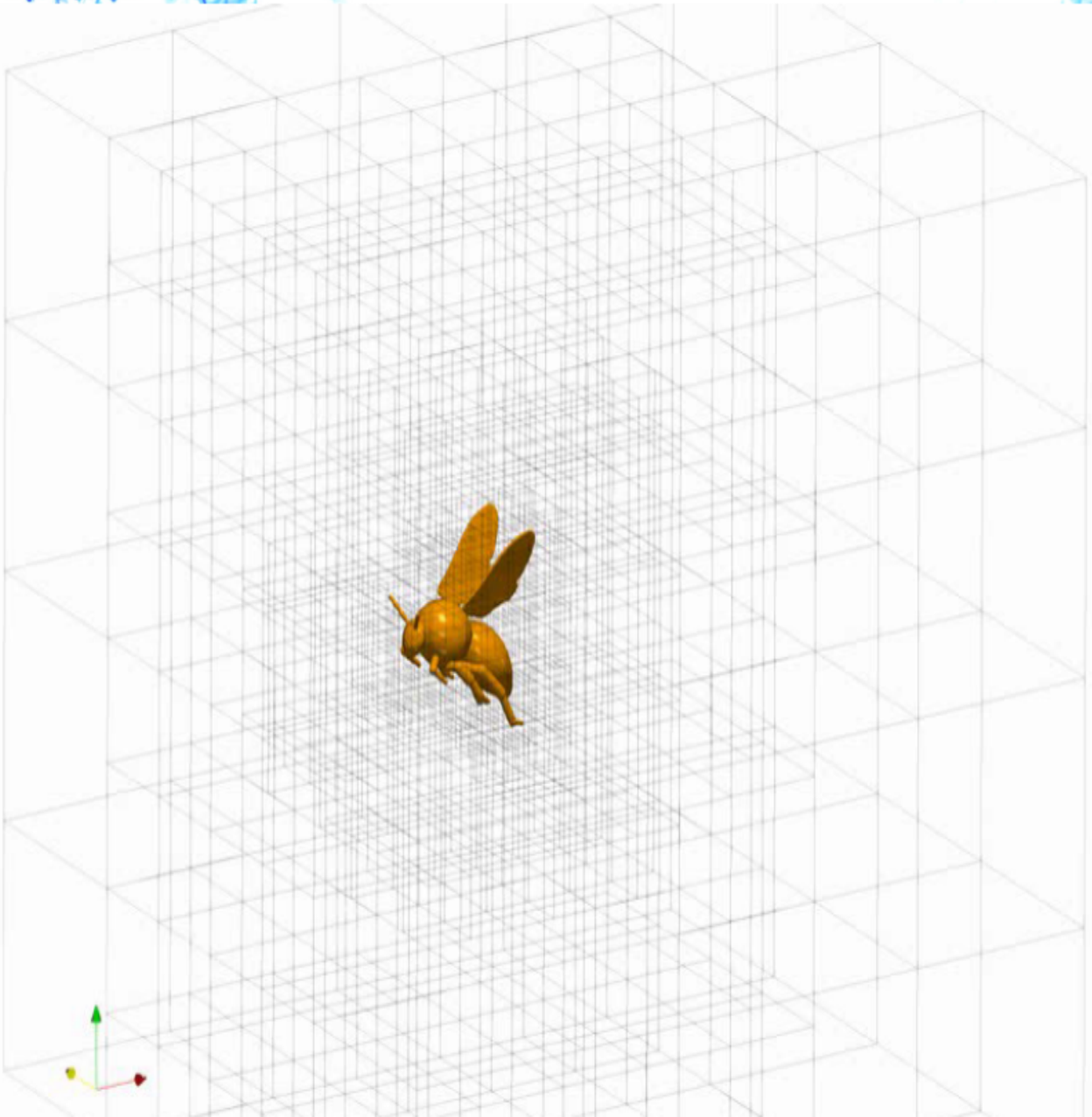
$$Re = 2060$$

$$Tu = u'/u_\infty = 0.0 - 0.99$$

Tethered or free flight (0 or 6 degrees of freedom)

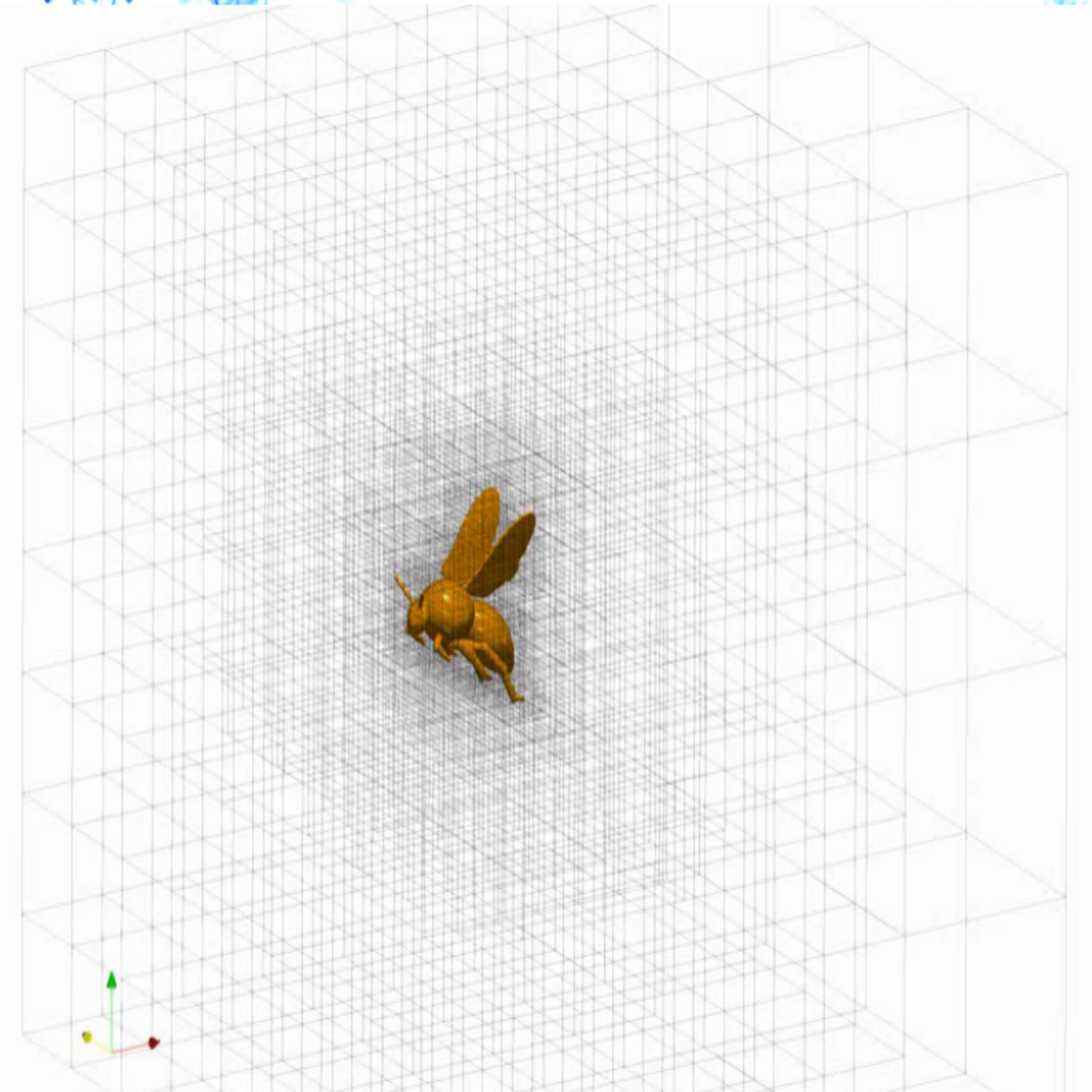
Engels et al. Bumblebee flight in heavy turbulence. Phys. Rev. Lett., **2016**

Flow and Grid (J=6)



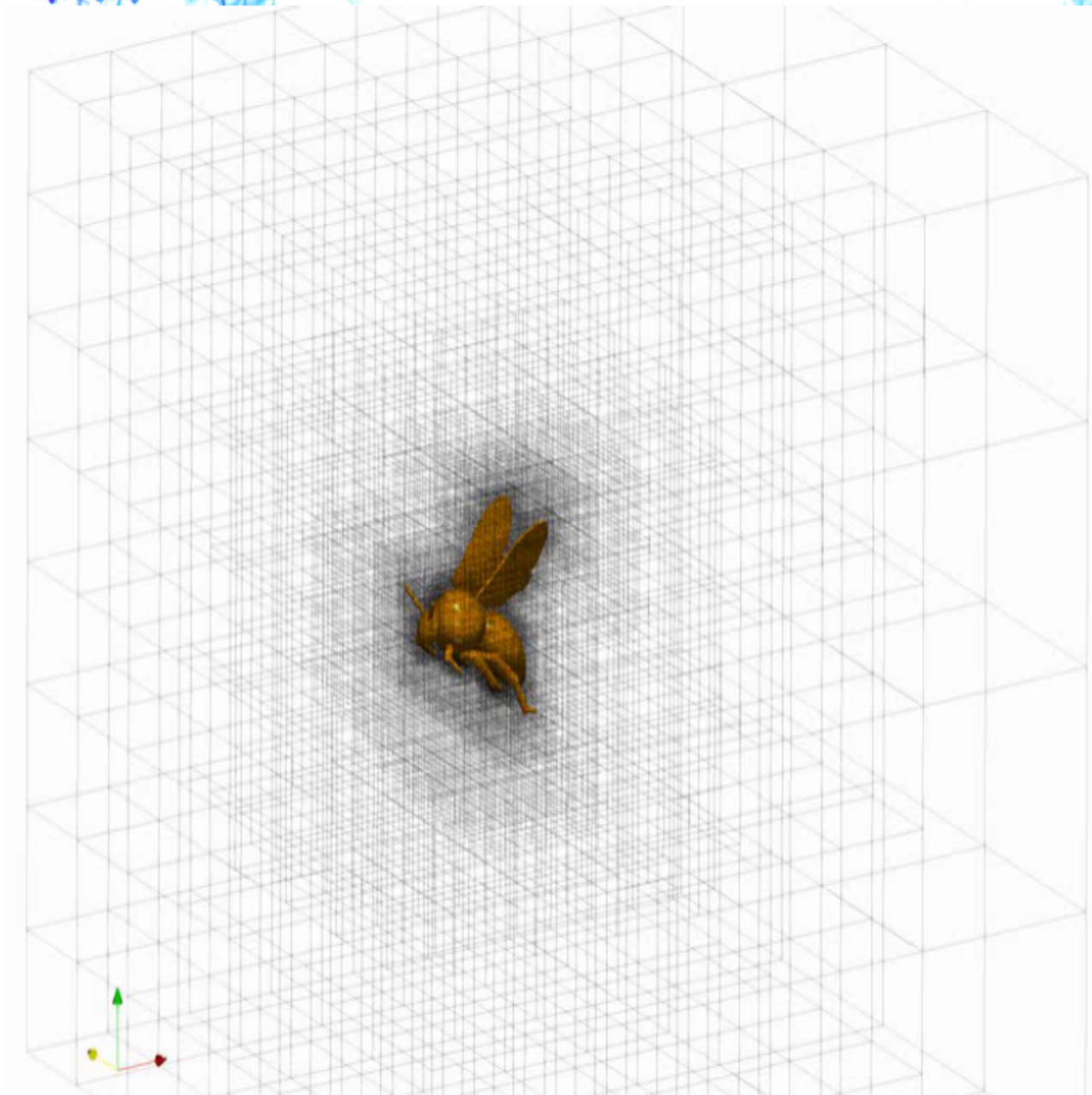
$$N_{\mathrm{CPU}} = 672$$

Flow and Grid (J=7)



$$N_{\mathrm{CPU}} = 672$$

Flow and Grid (J=8)



$N_{\mathrm{CPU}} < 1152$

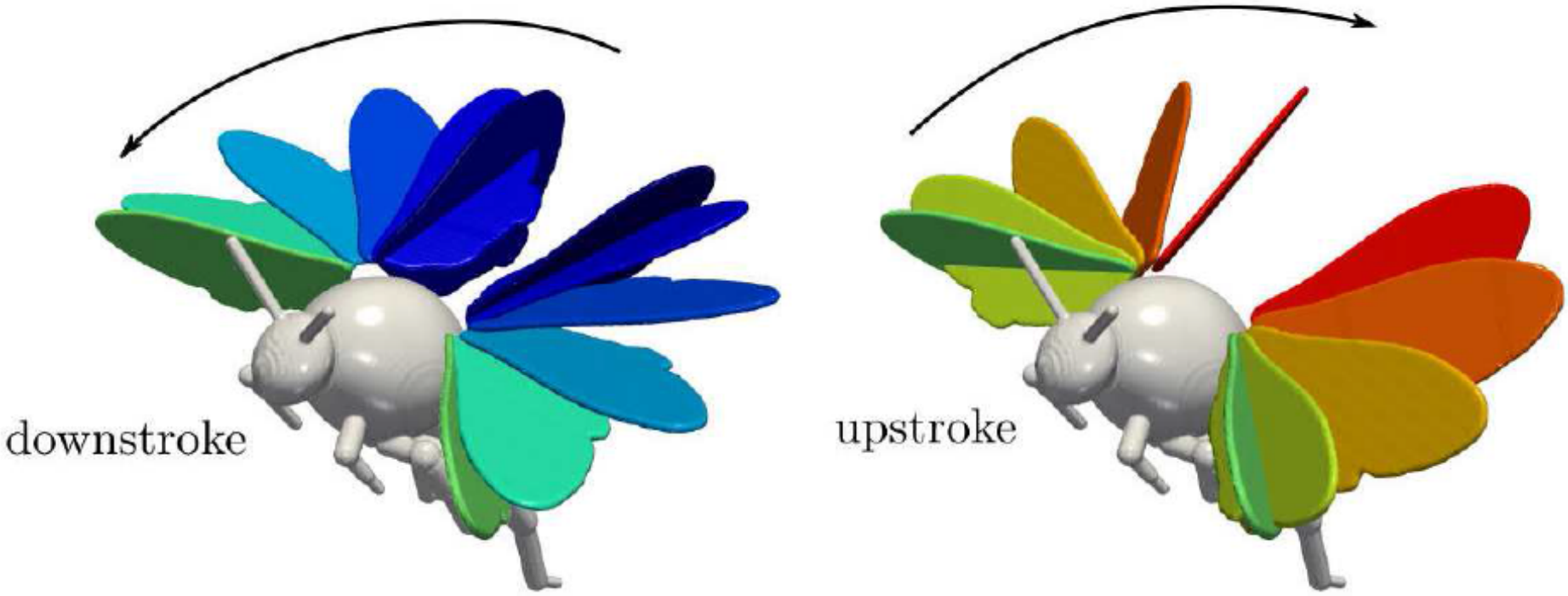
Performance

- Depending on the number of refinements level, speed up compared to spectral code increases
- Our new code is competitive with spectral method

Parameter	level 1	level 2	level 3
$J_{\max}$	6	7	8
C_ε	$4 \cdot 10^{-2}$	$1 \cdot 10^{-2}$	$2.5 \cdot 10^{-3}$
C_0	25	35	50
C_η	$5.45 \cdot 10^{-4}$	$1.36 \cdot 10^{-4}$	$3.14 \cdot 10^{-5}$
T_{CPU}	1 770	15 236	155 142
T_{CPU} (spectral)	26 858	560 0655	12 131 077
speedup	15.2	36.8	78.0

Bumblebee

Wingbeat of the bumblebee model in forward flight:



During the downstroke the angle of attack is more elevated than during the upstroke.

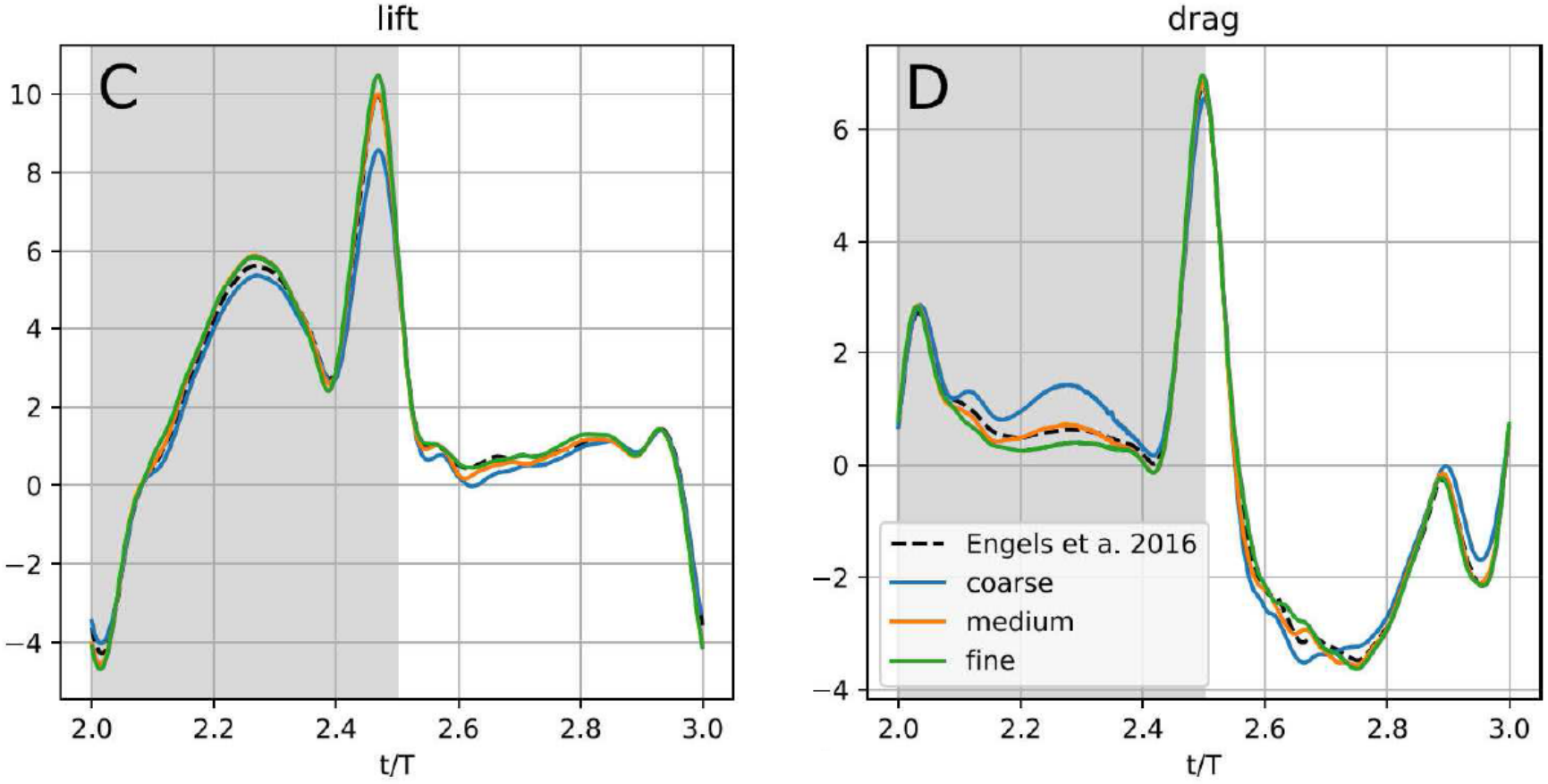
Bumblebee

Computational parameters for three levels of simulation.

	J_{max}	$h_{wing}/\Delta x$	C_ε	C_0	C_η	C_{sp}
coarse	6	4.4	$4 \cdot 10^{-2}$	25	$5.45 \cdot 10^{-4}$	$1.0 \cdot 10^{-3}$
medium	7	8.8	$1 \cdot 10^{-2}$	35.36	$1.36 \cdot 10^{-4}$	$7.1 \cdot 10^{-4}$
fine	8	17.6	$2.5 \cdot 10^{-3}$	50	$3.41 \cdot 10^{-5}$	$5 \cdot 10^{-4}$

Bumblebee

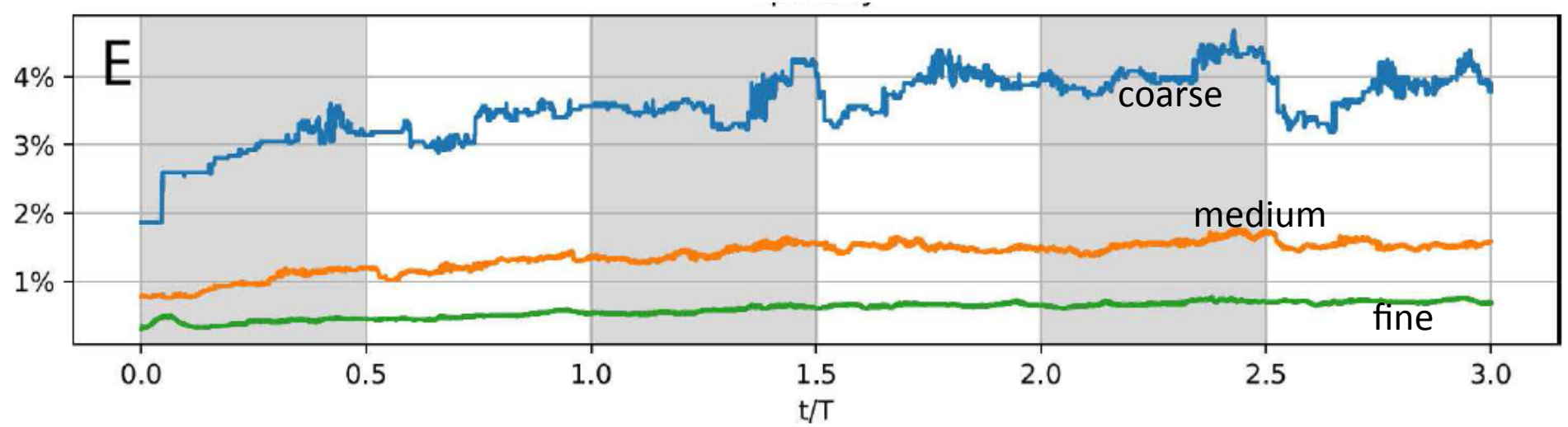
Lift and drag force.



Results obtained with CDF 4/0 wavelets.
Engels et al. 2016 are Fourier pseudo-spectral computations with Flusi.

Bumblebee

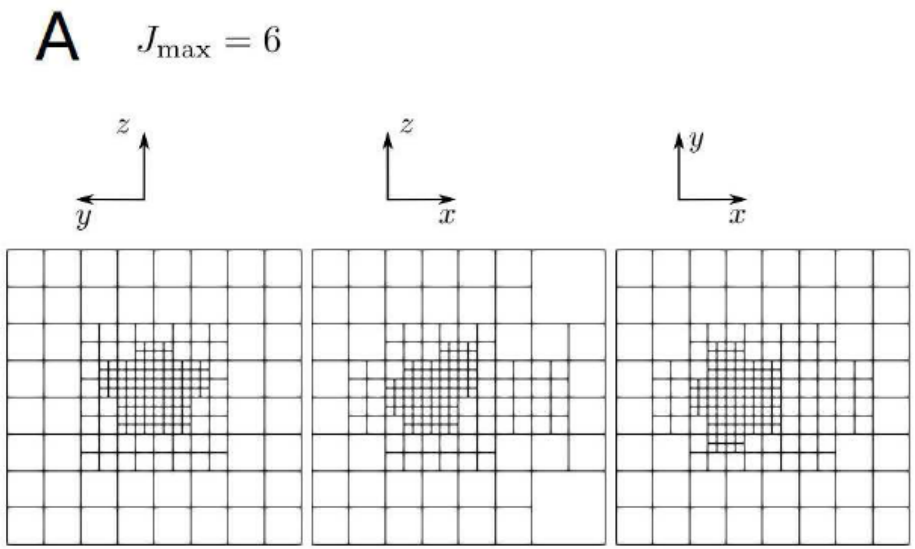
Grid sparsity as a function of time.



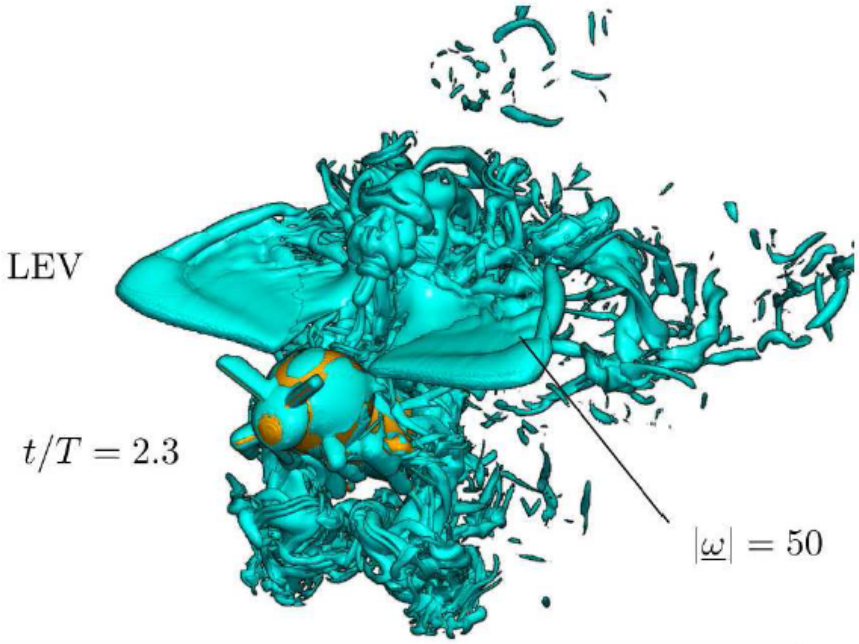
Results obtained with CDF 4/0 wavelets.

Bumblebee

Bumblebee simulations: coarse grid



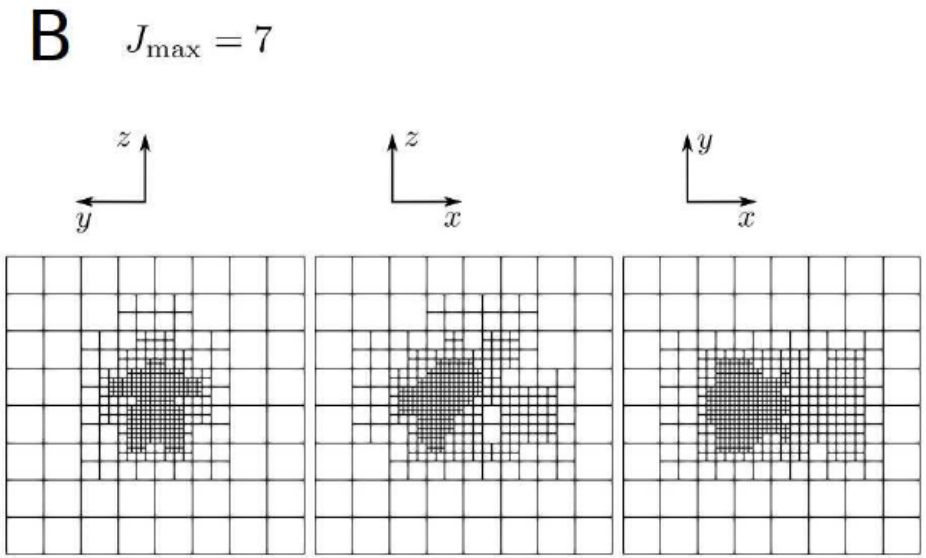
2D projection of the computational grid.



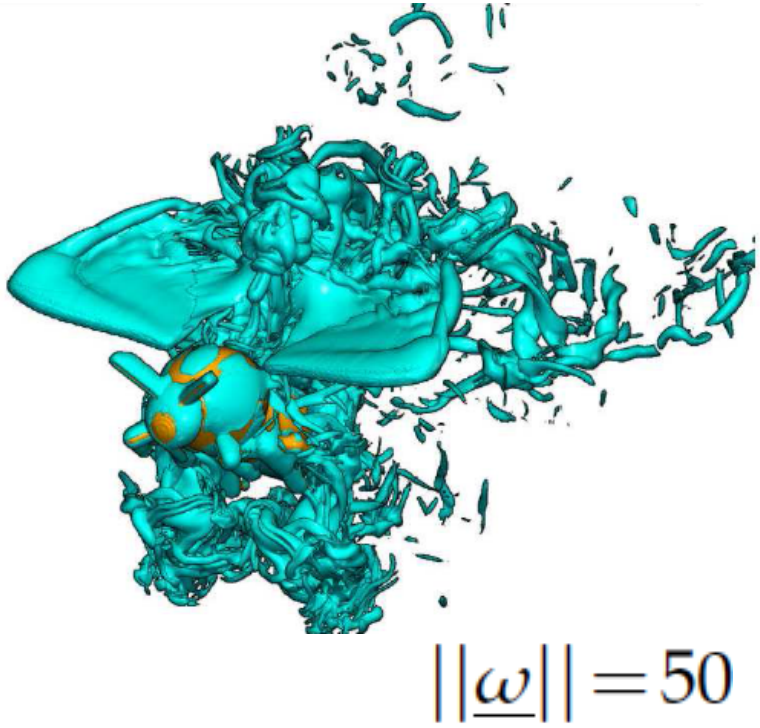
Visualization of the flow field.
Isosurface of vorticity magnitude.

Bumblebee

Bumblebee simulations: medium grid



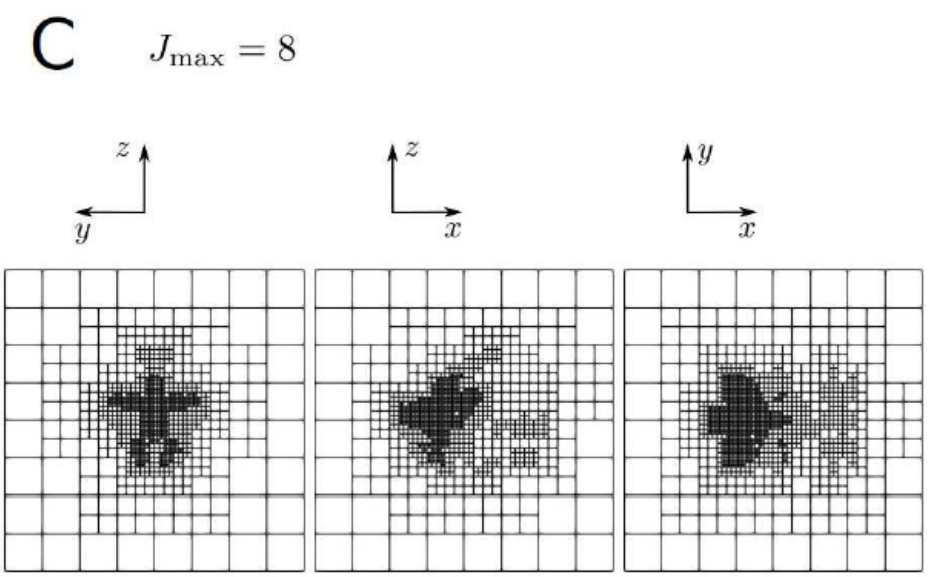
2D projection of the computational grid.



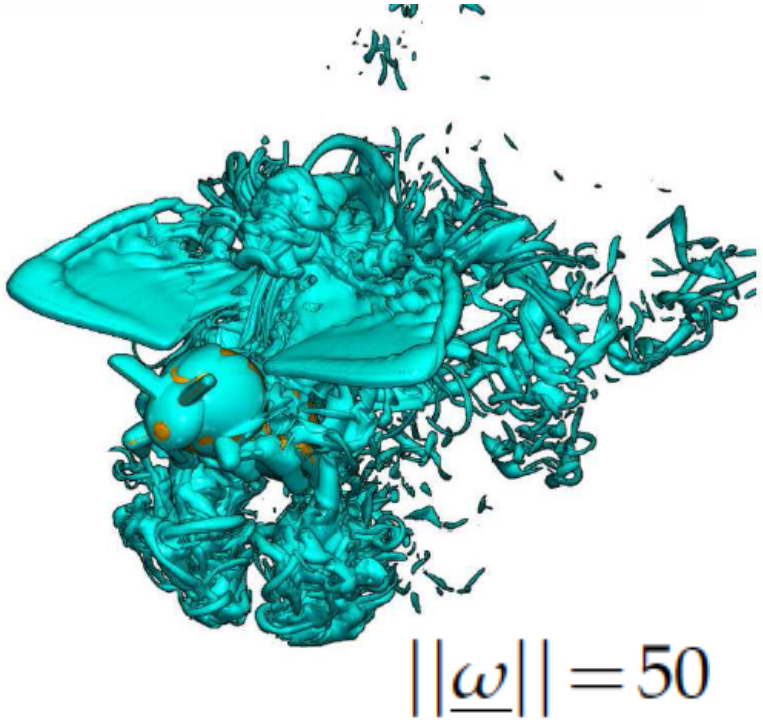
Visualization of the flow field.
Isosurface of vorticity magnitude.

Bumblebee

Bumblebee simulations: fine grid



2D projection of the computational grid.



Visualization of the flow field.
Isosurface of vorticity magnitude.

Computational performance

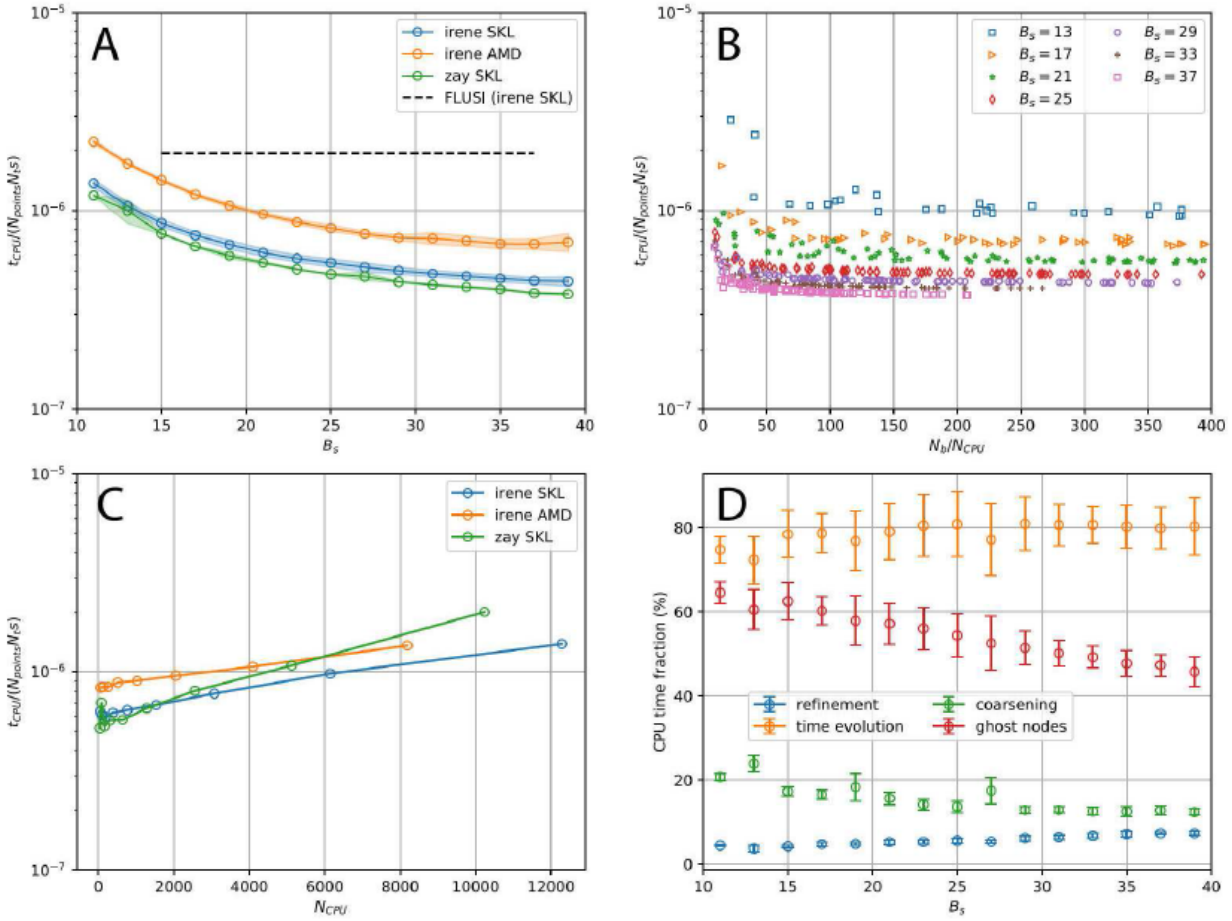


Figure 7: Computational performance on randomly generated 3D grids. A: CPU cost per grid point per right hand side evaluation as a function of B_s, measured on three supercomputers (`Irene SKL`, `Irene AMD`, `Zay SKL`). For comparison, cost using the pseudospectral `FLUSI` code on `Irene SKL` is included as well. Note only one simulation is shown, independent of B_s. Shaded areas correspond to mean $\pm$ 1 s.d. B: Cost as a function of N_b/N_{CPU} for different B_s, computed on $N_{\mathrm{CPU}} = 200$ cores on `Zay SKL`. C: for $B_s = 23$, cost as a function of N_{CPU} (weak scaling). For each datapoint load N_b/N_{CPU} is sufficiently high. D: fraction of cost spend on grid refinement, time evolution and coarsening (=100%), and ghost node synchronization via `MPI`. Computed on $N_{\mathrm{CPU}} = 200$ cores on `Zay SKL`. Results are obtained with CDF 4/0 wavelets and RK4 scheme ($s = 4$).

Computational performance

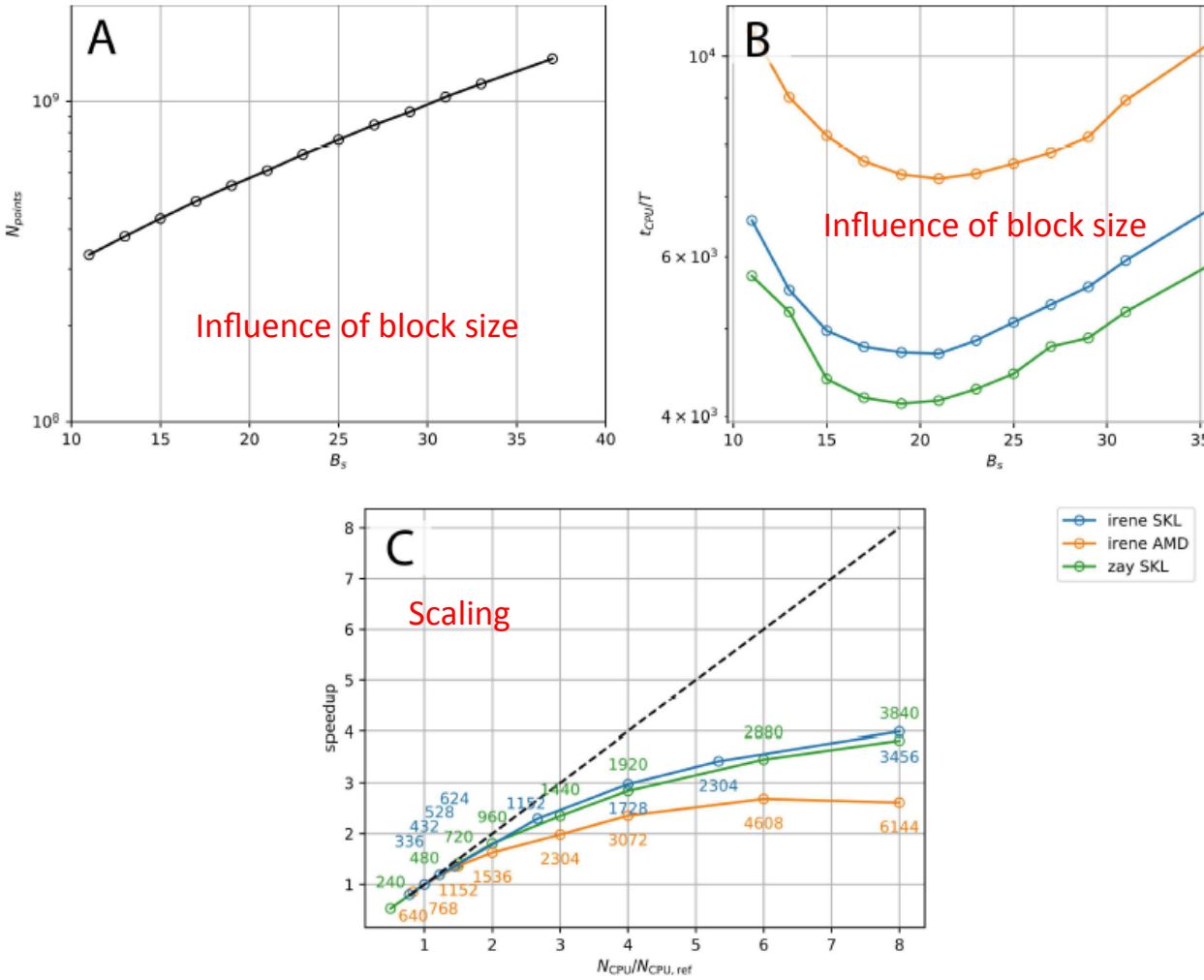


Figure 8: Computational performance in bumblebee simulations. A: time-averaged total number of points on the grid $N_{\text{points}} = N_b B_s^3$ as a function of B_s B: Total cost per time unit of a simulation with different B_s on different supercomputers. C: Strong scaling test on the 'fine' (cf Fig. 5) resolution simulation of the bumblebee on different supercomputers. For the simulated time span ($t = 0.1T$), N_b ranges from 49344 to 84176. Colored labels indicate N_{CPU}. Dashed line shows ideal scaling. Results are obtained with CDF 4/0 wavelets.

Bumblebee behind fractal tree

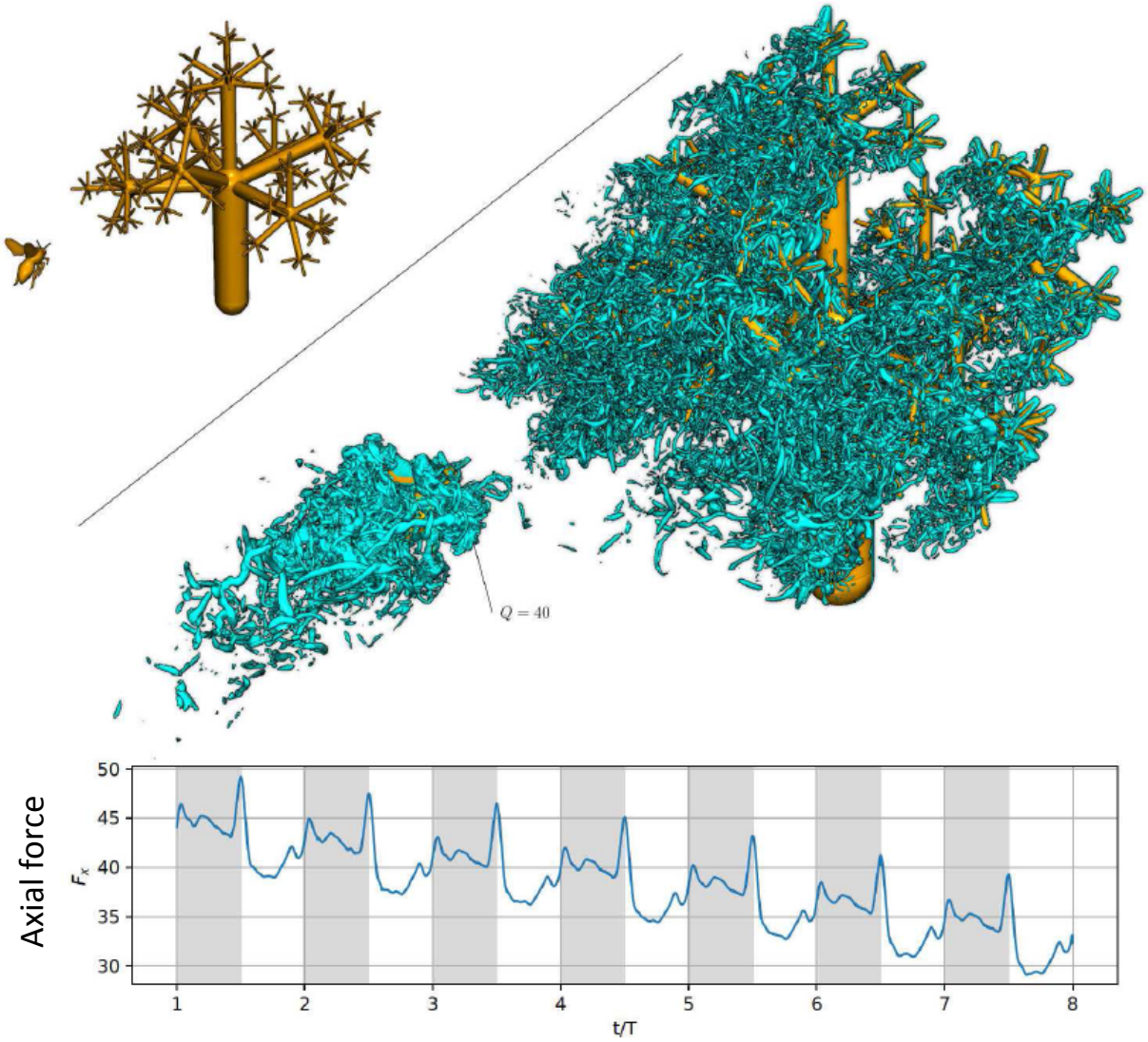
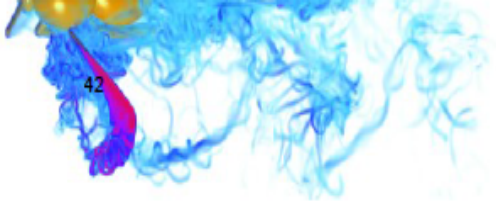
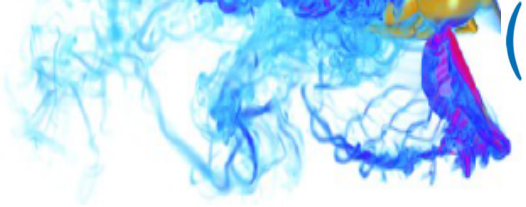


Figure 9: Bumblebee behind a fractal tree. Top part shows the setup consisting of a bumblebee and a tree-inspired fractal turbulence generator composed of rigid cylinders. Right part illustrates the flow field with an isosurface of the Q-criterion. Time in figure is $t/T = 8.0$, results obtained with CDF 4/0 wavelets. Bottom inset shows the axial force, *i.e.*, the force on the bumblebee acting in the direction of the fractal tree. Force decreases with time as the fractal tree's wake develops.



Conclusions



- We presented the open-source wavelet-adaptive code WABBIT, which can be used to solve PDEs on adaptive, block-based grids.
- Our physical model is based on artificial compressibility and volume penalization, which is a very general framework .
- The choice of parameters of the model has been discussed .
- The CPU cost is of the same order of magnitude as a Fourier pseudospectral code: savings in grid translate directly to CPU time and memory compression .
- Using the multiresolution threshold we can also perform simulations in the spirit of 'coherent-vortex simulation'. First results look promising.

Ref.: T. Engels, K. Schneider, J. Reiss and M. Farge. A wavelet-adaptive method for multiscale Simulation of turbulent flows in flying insects. *Commun. Comput. Phys.*, 30(4), 1118-1149, 2021.

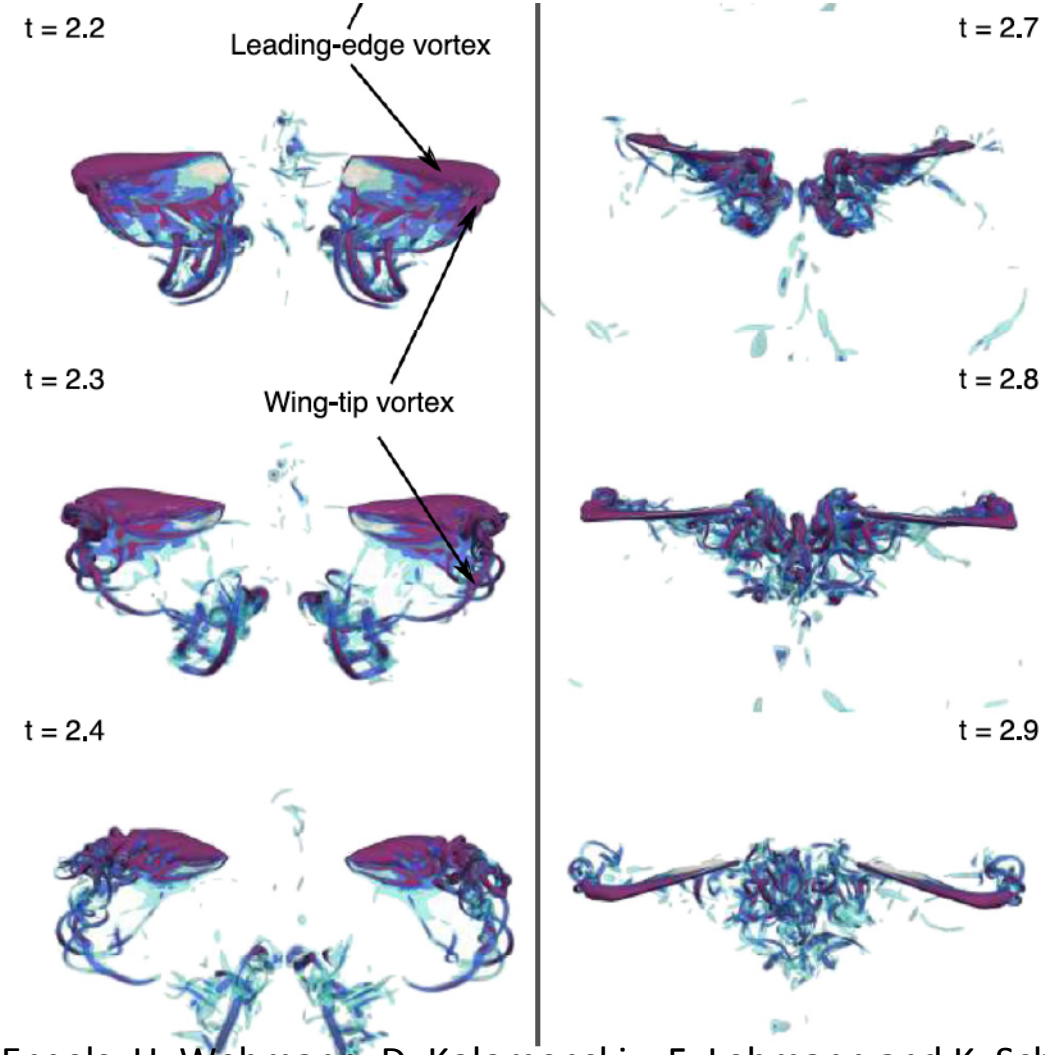
https://github.com/adaptive-cfd/WABBIT

Related publications, codes, videos etc can be found on

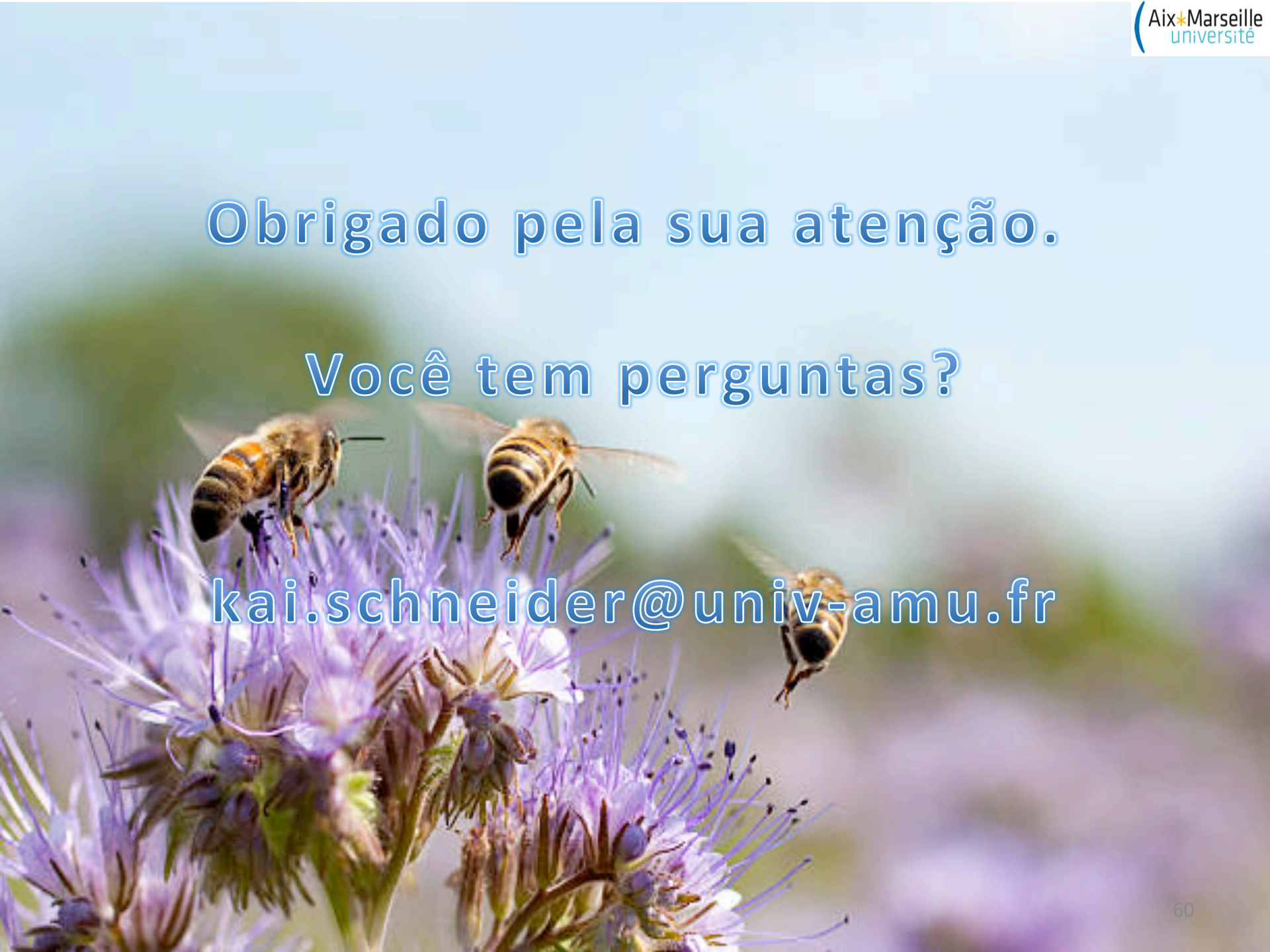
http://aifit.cfd.tu-berlin.de https://www.i2m.univ-amu.fr/perso/kai.schneider/

What next?

Adaptive computations of insects with flexible wings.



Ref.: H. Truong, T. Engels, H. Wehmann, D. Kolomenskiy, F. Lehmann and K. Schneider. An experimental data-driven mass-spring model of flexible Calliphora wings. *Bioinspiration & Biomimetics*, doi.org/10.1088/1748-3190/ac2f56, 2021, in press.



Obrigado pela sua atenção.

Você tem perguntas?

kai.schneider@univ-amu.fr