Commun. Comput. Phys.
doi: 10.4208/cicp.OA-2020-0246

Vol. **30**, No. 4, pp. 1118-1149
October 2021

A Wavelet-Adaptive Method for Multiscale Simulation of Turbulent Flows in Flying Insects

Thomas Engels[1,4,*], Kai Schneider[2], Julius Reiss[3] and Marie Farge[4]

[1] *Institute of Biosciences, University of Rostock, Rostock, Germany.*
[2] *Aix-Marseille Université, CNRS, I2M UMR 7373, Marseille, France.*
[3] *ISTA, Technische Universität Berlin, Berlin, Germany.*
[4] *LMD UMR 8539 École Normale Supérieure-PSL, Paris, France.*

Received 4 December 2020; Accepted (in revised version) 7 May 2021

Abstract. We present a wavelet-based adaptive method for computing 3D multiscale flows in complex, time-dependent geometries, implemented on massively parallel computers. While our focus is on simulations of flapping insects, it can be used for other flow problems. We model the incompressible fluid with an artificial compressibility approach in order to avoid solving elliptical problems. No-slip and in/outflow boundary conditions are imposed using volume penalization. The governing equations are discretized on a locally uniform Cartesian grid with centered finite differences, and integrated in time with a Runge–Kutta scheme, both of 4th order. The domain is partitioned into cubic blocks with different resolution and, for each block, biorthogonal interpolating wavelets are used as refinement indicators and prediction operators. Thresholding the wavelet coefficients allows to generate dynamically evolving grids, and an adaption strategy tracks the solution in both space and scale. Blocks are distributed among MPI processes and the grid topology is encoded using a tree-like data structure. Analyzing the different physical and numerical parameters allows us to balance their errors and thus ensures optimal convergence while minimizing computational effort. Different validation tests score accuracy and performance of our new open source code, `WABBIT`. Flow simulations of flapping insects demonstrate its applicability to complex, bio-inspired problems.

AMS subject classifications: 65M06, 65M50, 65M55, 65M85, 65T60, 76M20, 76D05, 76Z10

Key words: Adaptive numerical method, wavelets, volume penalization, artificial compressibility, flapping flight.

---

[*] Corresponding author. *Email addresses:* `thomas.engels@ens.fr` (T. Engels), `kai.schneider@univ-amu.fr` (K. Schneider), `julius.reiss@tnt.tu-berlin.de` (J. Reiss), `marie.farge@ens.fr` (M. Farge)

1 Introduction

Computing multiscale flows in complex geometries, which may move or deform, is required in numerous applications, *e.g.*, many biological flow problems such as flying insects or beating hearts. This remains a major challenge for computational fluid dynamics, especially in the turbulent flow regime. To simulate turbulent flows in the presence of moving boundaries, numerical techniques are needed that allow the solution to be tracked in both space and scale, and the numerical grid to be dynamically adapted accordingly. If the fluid-structure interaction must also be taken into account, it is all the more difficult since the motion of the boundary is no longer known *a priori*, but depends on its nonlinear interaction with the fluid. In this regard adaptive numerical discretization methods, which can be traced back to the 1980s [6,7], are indeed attractive. In many cases they can be much more competitive than schemes on uniform fine grids, depending on the character of the solution. However, for adaptive discretizations two major challenges can be identified: their actual implementation on massively parallel supercomputers and the numerical error analysis of adaptivity.

The implementation of the code is crucial to optimize computing, and two conceptually different approaches can be distinguished: one uses point-based techniques, while the other uses block-based techniques. In the former, the error indicator determines for each grid point whether it is significant or not (*e.g.*, [45,58]), while in the latter significant grid points are clustered in patches with the drawback of including non-significant points and thus decreasing the compression rate [24]. Due to the hardware layout of modern CPU, block-based implementations are in many cases more competitive and have become increasingly popular during the last years, see, *e.g.*, [62] for a recent review on available software packages. The locally regular block data can be transferred in one contiguous chunk to the CPU cache, which greatly increases the performance despite an increase in the number of points.

The mathematical support for adaptivity needs to provide reliable error estimators of the solution and, for evolutionary problems, a prediction of the grid used to compute the next time step. For both, a variety of heuristic criteria exists, *e.g.*, gradient-based approaches [18]. Adaptive mesh refinement algorithms use these heuristic criteria and are *error-indicated* methods because of their heuristic nature. *A posteriori* error estimators [4] are mathematically rigorous but require solving expensive adjoint problems.

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*. The idea of wavelet analysis is to decompose data into contributions in both space and scale (and possibly direction). The wavelet transform has been introduced by Grossmann and Morlet [37], and the algorithm of the fast wavelet transform by Mallat [49]. Nonlinear approximation [21] provides the conceptual support for adaptivity; indeed, it introduces a systematic way to classify functions according to the sparsity of their representation in wavelet space, *i.e.*, the possibility of describing a function by a small number of wavelet coefficients. Wavelet-based algorithms for solving PDEs, starting with [48], were the first ones to be proved to guarantee the best n-term approximation [11]. The books of Cohen [10] and Müller [52] and the review articles [23, 61] give a detailed overview on the subject.

To take into account boundary conditions for complex geometries, in particular at solid walls, the family of immersed boundary methods (IBM) [51, 56, 60] is now often preferred over the traditional body-fitted grids for complex geometries. While the latter can yield better accuracy near the wall, IBM approaches convince with their ease of implementation and great flexibility. An early application for computing flow in a beating heart can be found in [55], illustrating that this class of methods has been designed from the start for complex, bio-inspired flow problems. Among the variety of methods, the volume penalization method [3], physically motivated by flow in porous media, is furthermore distinguished by its convergence proof for nonlinear Navier–Stokes [2]. Moreover, its convergence properties are improved in the turbulent regime [53].

In biological applications the flow can be typically described as incompressible, which requires specialized numerical techniques to decouple the pressure from the velocity, most of which involve solving a Poisson equation. While there are adaptive solvers following this strategy, *e.g.*, [45], alternative methods such as the Lattice–Boltzmann method are an attractive choice for adaptive simulations due to their explicit character [62]. Another recently revisited method [54] is the artificial compressibility method [9], which introduces a large but finite speed of sound and avoids solving elliptic equations.

In this work, we present a wavelet-based fully adaptive approach for solving the incompressible Navier–Stokes equations in complex and time-varying geometries on massively parallel computers. To this end, we combine two physically motivated models: the artificial compressibility method, which relaxes the kinematic incompressibility constraint and allows for a finite speed of sound, and the volume penalization, which models solid objects as porous media with finite permeability. In the limit when the speed of sound and the permeability parameters tend to infinity and to zero, respectively, we recover the incompressible Navier–Stokes equations with no-slip boundary conditions. The numerical method for solving the resulting model equations is based on centered finite differences and explicit time integration, both of 4th order. The MPI parallel implementation uses block-based grids, which are coarsened and refined using biorthogonal wavelets. While the framework we present is quite general and can be applied to a large variety of fields, our main motivation for its development comes from the spectacular flight capabilities of flying insects, which use flapping wings to produce the required aerodynamic forces.

The remainder of the paper presents the governing equations (Section 2), followed by the numerical method including discretization errors (Section 2.3), and the multiresolution analysis for introducing adaptivity (Section 3). The choice of the parameters for balancing the different errors is discussed in Section 3.6. The parallel implementation is presented in Section 4. Different numerical results are shown in Section 5, and the performance of the code is analyzed in Section 5.4. Finally, an outlook is given in Section 6 conclusions are drawn in Section 7.

2 Governing equations and numerical method

2.1 Artificial compressibility

The applications we have in mind involve exterior, incompressible flows past obstacles of complex shape which may vary in time. Numerical methods for such flows usually require the solution of a Poisson-type, elliptic PDE. Even though powerful libraries (*e.g.*, PETSc [1]) exist for that purpose, the solution of the Poisson equation is causing major computational cost, especially on irregular, time-dependent grids.

Instead of the incompressible Navier–Stokes equations,

$$\partial_t \underline{u} + (\underline{u} \cdot \nabla) \underline{u} + \nabla p - \nu \nabla^2 \underline{u} = 0, \tag{2.1}$$

$$\nabla \cdot \underline{u} = 0 \tag{2.2}$$

for the velocity $\underline{u}$ and the pressure p in the fluid domain $\Omega_f \subset \mathbb{R}^D$ for $t>0$ and $D=2$ or 3, completed with suitable boundary and initial conditions, and where ν is the kinematic viscosity, we consider the artificial compressibility method (ACM). An artificial speed of sound $C_0 \gg |\underline{u}|$ is introduced and Eq. (2.2) is replaced by

$$\partial_t p + C_0^2 \nabla \cdot \underline{u} + C_\gamma p = 0, \tag{2.3}$$

where $C_\gamma > 0$ is a dashpot damping term discussed below. Note that the fluid density ϱ_f is normalized to unity and the equations are written in dimensionless form. The basic idea of the ACM can be traced back to Chorin [9]. More recently, several authors have reconsidered it. Guermond and Minev state that “the artificial compressibility method may not have been given all the attention it deserves in the literature” [38]. The spirit adopted here, which is to explicitly solve Eqs. (2.1), (2.3), is inspired by the work of Ohwada and Asinari [54]. The ACM solution converges to the solution of the incompressible Navier–Stokes equations (INC) with

$$e_{\mathrm{ACM}} = \|\underline{V}_{\mathrm{ACM}} - \underline{V}_{\mathrm{INC}}\|_2 = \mathcal{O}(C_0^{-2}) \tag{2.4}$$

as $C_0 \to \infty$. Here, $\underline{V}$ is the state vector, containing $\underline{u}$ and p. We will confirm this result numerically later. Note that [54] proposes a strategy to increase the accuracy to $\mathcal{O}(C_0^{-4})$ using Richardson extrapolation, but we do not follow this approach because the second order is sufficient when combined with the volume penalization method. We also point out that there is a natural upper limit for C_0, which is the true speed of sound of the fluid (*e.g.* $C_0 = 340\mathrm{ms}^{-1}$ in air), and that any fluid is at least slightly compressible.

2.2 Volume penalization

The no-slip boundary conditions at the fluid–solid interface are imposed using the volume penalization method [2,3], which is based on the physical intuition of approximating a solid obstacle as a porous medium with small permeability. The penalized versions of Eqs. (2.1), (2.3) are solved in the larger domain $\Omega=\Omega_f \bigcup \Omega_s$, where Ω_s is the solid domain, and read

$$\partial_t \underline{u} + (\underline{u}\dot{\nabla})\underline{u} + \nabla p - \nu \nabla^2 \underline{u} + \frac{\chi}{C_\eta}(\underline{u}-\underline{u}_s) + \frac{\chi_{\mathrm{sp}}}{C_{\mathrm{sp}}}(\underline{u}-\underline{u}_\infty) = 0, \tag{2.5}$$

$$\partial_t p + C_0^2 \nabla\cdot\underline{u} + C_\gamma p + \frac{\chi_{\mathrm{sp}}}{C_{\mathrm{sp}}}(p-p_\infty) = 0. \tag{2.6}$$

Here χ is the indicator function, $C_\eta \ll 1$ is the penalization parameter (permeability) and $\underline{u}_s$ the solid body velocity field. The indicator function is $\chi=1$ inside Ω_s and $\chi=0$ in Ω_f, and we use a thin smoothing layer with thickness proportional to the grid spacing in the case of moving [47] or deforming obstacles [31]. Let $\delta(\underline{x})$ be the signed distance to the fluid-solid interface, then

$$\chi(\delta) = \begin{cases} 1, & \delta \leq -h, \\ \frac{1}{2}\left(1+\cos\left(\pi \frac{\delta+h}{2h}\right)\right), & -h < \delta < +h, \\ 0, & \delta > +h, \end{cases} \tag{2.7}$$

where $h = C_{\mathrm{smooth}} \Delta x$ is the smoothing layer thickness and $C_{\mathrm{smooth}} = 1.0$. The adaptive grid is always refined to the highest resolution at the fluid-solid interface, and hence $\Delta x = \Delta x_{\mathrm{min}}$ (see Section 3.5). For details on the generation of χ, see [33]. From the penalization term aerodynamic forces are obtained by volume integration [2],

$$\underline{F} = C_\eta^{-1} \int_\Omega \chi(\underline{u}-\underline{u}_s)\, \mathrm{d}\Omega,$$

for which we use the midpoint quadrature rule. We anticipate that the penalization error converges as

$$e_{\mathrm{VPM}} = \left\| \underline{V}_{\mathrm{ACM},C_\eta} - \underline{V}_{\mathrm{ACM}} \right\|_2 = \mathcal{O}(C_\eta^{1/2}) \tag{2.8}$$

for $C_\eta \to 0$ using similar arguments as for INC.

For our application to unbounded exterior flows, an additional sponge penalization term (index sp) has been added [31, 32], which imposes mean flow conditions, $\underline{u}_\infty$ and p_∞, in a layer at the outer boundary of the computational domain. Contrarily to the penalization for the obstacle itself, this term also acts on the pressure p. We can understand the ACM as a transport of divergence, which is created both at the boundary and in the nonlinear term. Hence, some outflow conditions are required to remove it from the physical domain, and the sponge term here plays the dual role of absorbing outgoing pressure waves as well as damping outgoing vortical structures.

In the case of impulsively started flow, the initial condition is incompatible with the boundary conditions, which causes a singularity in the aerodynamic force. A strong pressure wave of width $\delta_s = \mathcal{O}(C_\eta C_0)$ is created at the boundary, because the penalization requires a time $\mathcal{O}(C_\eta)$ to damp the velocity inside the obstacle, and the ACM transports it with a speed C_0. Its intensity decays with the distance R as $R^{-(D-1)}$, where $D=2$ or 3 is the dimensionality of the problem. Even though, physically speaking, the pressure wave is an artifact of the ACM. It still needs to be resolved, as it transports information needed to create an (approximative) solenoidal flow field in the full domain. If δ_s is too small, dispersion errors occur, and spurious oscillations in the forces can appear. In practice, we ensure that $\Delta x \leq \delta_s$. In order to minimize reflections, the width of the sponge layer L_{sp} is set larger than characteristic length scales of the geometry; a wider sponge reduces reflections. We then adjust the sponge constant C_{sp} such that the time the pressure wave spends in the sponge is equivalent to τ times the relaxation time, yielding $C_{sp} = L_{sp}(C_0\tau)^{-1}$. Here we set $\tau = 20$, but found that the specific choice of both τ and L_{sp} only has a weak impact on the solution (see Appendix D in suppl. material for details).

In simulations with periodic boundary conditions, the sponge technique cannot be used. For such cases, we use the dashpot damping term $C_\gamma p$ proposed in [54] to mitigate pressure waves; the constant is then $C_\gamma = 1$, as suggested in [54].

The combination of artificial compressibility and volume penalization constitutes our physical model. It has four parameters (C_η, C_0, C_{sp}, C_γ), the choice of which will be discussed in detail below.

2.3 Numerical method and discretization errors

The model equations (2.5)-(2.6), are discretized in space and time. Spatial derivatives are approximated using centered higher-order finite differences on a collocated, periodic grid. We use the optimized fourth order scheme proposed by Tam and Webb [67] for first order derivatives (Appendix A in suppl. material) and a classical, centered, fourth order stencil for second order derivatives.

Time integration is done with a classical Runge–Kutta–4 scheme (RK4). The explicit scheme implies stability restrictions on the time step Δt, namely

$$\Delta t < \min\left(\mathrm{CFL}\Delta x_{\min}/u_{\mathrm{eig}}, \mathrm{CFL}_\eta C_\eta, \mathrm{CFL}_\nu \Delta x_{\min}^2/\nu\right),$$

where the characteristic velocity is $u_{\mathrm{eig}} = |u_{\max}| + \sqrt{u_{\max}^2 + C_0^2}$, which we can approximate by $u_{\mathrm{eig}} \approx C_0$ as $C_0 \gg |u_{\max}|$. The constants are set to $\mathrm{CFL} = \mathrm{CFL}_\eta = 1.0$ and $\mathrm{CFL}_\nu = 1/4$. We therefore expect the fully discretized solution to converge with 4th order accuracy in space and time,

$$e_{\mathrm{FDM}} = \left\| \underline{V}_{\mathrm{ACM}}^{\Delta x, \Delta t} - \underline{V}_{\mathrm{ACM}} \right\|_2 = \mathcal{O}(\Delta x^4) + \mathcal{O}(\Delta t^4), \tag{2.9}$$

which we will verify later numerically. This is valid in the case without penalization. As pointed out in [53], the discretization (Eq. (2.9)) and penalization (Eq. (2.8)) error cannot be treated independently. A too small value for C_η results in a loss of regularity of the exact solution and by consequence the discretization error increases, thus there is an optimal choice of C_η as a function of Δx. As discussed in our previous work [32, 33], the relation is

$$C_\eta = \left(K_\eta^2/\nu\right)\Delta x^2, \tag{2.10}$$

where K_η is a constant which depends only on the discretization scheme. It can be interpreted as the number of grid points in the penetration boundary layer inside the obstacle. This relation is also the reason for our choice of time stepping method (explicit RK4). It is well adapted to the convection-based, turbulent problems we are interested in because the constraint $\Delta t < C_\eta$ is less severe in such cases. Simulations with large viscosity may require different time steppers, such as Runge–Kutta–Chebychev methods, which are also implemented in the `WABBIT` code but not the focus of present work. Using Eq. (2.10), first order convergence [53] is assured for any value of K_η, *i.e.*,

$$\left\|\underline{V}_{\mathrm{ACM},C_\eta}^{\Delta x,\Delta t} - \underline{V}_{\mathrm{ACM}}\right\|_2 = \mathcal{O}(\Delta x) + \mathcal{O}(\Delta t^4), \tag{2.11}$$

as $\Delta x, C_\eta \to 0$. The error offset can be tuned by choosing K_η. In the present work, we obtained best results using the same values of $0.05 \le K_\eta \le 0.4$ as reported in [33] with a Fourier discretization, which suggests a certain universality of this scaling.

3 Multiresolution analysis and grid adaptation

Motivated by the multitude of scales which are excited in the flow, especially in the turbulent regime, and close to the boundary, whose location may even change in time, we introduce dynamically adaptive grids. The grid is supposed to track automatically the motion in both scale and position and allows furthermore its prediction for the next time step. To this end we use multiresolution analysis and decompose the solution into biorthogonal wavelets. First we recall the point-value multiresolution analysis and we give the link with biorthogonal wavelets. Some drawbacks of point-value multiresolution analysis are pointed out and we show how they can be overcome using lifted wavelets. Some details on the parallel implementation are given later in Section 4.

For ease of presentation we limit the description to the one-dimensional scalar-valued case, denoted by v. The extension to higher dimension and vector-valued data is outlined at the end of the section.

3.1 Point-value multiresolution analysis

Discrete multiresolution analysis introduced by Harten [39–41] is well-suited for introducing adaptivity into discretization schemes of PDEs. In particular for finite difference methods the point-value multiresolution [39], which is directly related to the interpolating subdivision scheme by Deslauriers & Dubuc [19, 20], has been designed.

Starting point is a nested sequence of uniform grids at level J, $X^J=\{x_i^J\}_{i=0}^{2^J}$ with an odd number of grid points defined by $x_i^J=i/2^J$ for $i=0,\cdots,2^J$. The grids satisfy $x_{2i}^{J+1}=x_i^J$ for $i=0,\cdots,2^J$ and $x_{2i-1}^{J+1}=(x_i^J+x_{i-1}^J)/2$ for $i=1,\cdots,2^J$.

For the solution v given on the grid X^J we can then define a prediction operator $P_{J\to J+1}: v(X^J)\to v(X^{J+1})$ which interpolates values of v onto the next finer grid. For this, we use the fourth order interpolating subdivision scheme [19,20] to interpolate missing values at x_{2i+1}^{J+1}, *i.e.*,

$$v_{2i+1}^{J+1}=-\frac{1}{16}v_{i-1}^J+\frac{9}{16}v_i^J+\frac{9}{16}v_{i+1}^J-\frac{1}{16}v_{i+2}^J. \tag{3.1}$$

Following Harten, we also define a restriction operator $R_{J+1\to J}: v(X^{J+1})\to v(X^J)$ which is simply the decimation by a factor of two, *i.e.*, $v_i^J=v_{2i}^{J+1}$. Note that $R_{J+1\to J}\circ P_{J\to J+1}=\mathrm{Id}$. Combining restriction and prediction, detail coefficients can be computed by subtracting the values predicted from the coarse grid from the values at the fine grid. At even grid points the details are zero, while at odd grid points we obtain

$$\widetilde{v}_{2i+1}^J=v_{2i+1}^{J+1}-P_{J\to J+1}R_{J+1\to J}v_{2i+1}^{J+1}. \tag{3.2}$$

Thus, the solution on the fine grid v^{J+1} can be represented as a solution at the coarser grid v^J plus detail coefficients $\widetilde{v}^J$. Iterating this procedure from J down to 1 yields the multiresolution transform. The fine grid solution can be reconstructed by inverting the above procedure. Detail coefficients are mostly significant in regions of steep gradients and discontinuities, and are small or even do vanish in regions where the solution is smooth. They "measure the local deviation from a polynomial" and are obtained as the interpolation error. Thus adaptivity can be introduced by removing small detail coefficients without loosing the precision of the computation.

3.2 Biorthogonal wavelets

The point-value multiresolution analysis is intimately related to biorthogonal wavelets, as already pointed out by Harten [39], and the solution $v\in C^s(\mathbb{R})$ $(s\geq 0)$ can be expanded into a biorthogonal wavelet series,

$$v(x)=\sum_{i\in\mathbb{Z}}\left\langle v,\widetilde{\phi}_i^0(x)\right\rangle\phi_i^0(x)+\sum_{J=0}^{\infty}\sum_{i\in\mathbb{Z}}\left\langle v,\widetilde{\psi_i^J}(x)\right\rangle\psi_i^J(x). \tag{3.3}$$

Note that for a given primary scaling function ϕ, different dual scaling functions $\widetilde{\phi}$ can be constructed (and vice versa) and thus the choice is not unique and there is some flexibility compared to the orthogonal case (in terms of symmetry, number of vanishing moments, filter length, etc.). For orthogonal wavelets we have $\widetilde{\phi}=\phi$ and $\widetilde{\psi}=\psi$.

For the point-value multiresolution the scaling function ϕ (cf. Appendix C in suppl. material) is identical to the Deslauriers–Dubuc fundamental function [19,20]. It can be obtained by interpolating a delta Dirac impulse (using Lagrange interpolation), and the filters are defined to be symmetrical and have minimal support for a given polynomial order. They are equivalent to the autocorrelation function of orthogonal Daubechies [14] scaling functions [50]. The dual scaling functions are delta Dirac distributions $\widetilde{\phi}_k^J(x) = \delta(x-x_k^J)$. The wavelets are interpolating ('interpolets'), yield an interpolating subdivision scheme, and correspond to a shifted scaling function $\psi(x)=\phi(2x-1)$, while the dual wavelets are linear combinations of δ distributions. The construction of these interpolating wavelets was also proposed by Donoho [25] and rediscussed later by Sweldens [65]. Note that the primary scaling function is normalized with respect to the L^∞ norm, *i.e.*, $\phi_i^J(x)=\phi(2^J x-i)$ implying $\left\|\phi_i^J\right\|_\infty=1$, and not with respect to the L^2 norm, as it is typically the case.

Both scaling functions and wavelets fulfill refinement relations,

$$\phi(x)=\sum_{n\in\mathbb{Z}} h_n\phi(2x-n), \quad \widetilde{\phi}(x)=\sum_{n\in\mathbb{Z}} \widetilde{h}_n\widetilde{\phi}(2x-n), \tag{3.4}$$

$$\psi(x)=\sum_{n\in\mathbb{Z}} g_n\phi(2x-n), \quad \widetilde{\psi}(x)=\sum_{n\in\mathbb{Z}} \widetilde{g}_n\widetilde{\phi}(2x-n), \tag{3.5}$$

and the filter coefficients are coupled via $\widetilde{g}_n=(-1)^{1-n}h_{1-n}$ and $g_n=(-1)^{1-n}\widetilde{h}_{1-n}$.

Rearranging the restriction and prediction operations of the multiresolution transform yields the filter coefficients. Then applying the fast wavelet transform with these filters and downsampling will yield equivalent results. The detail coefficients correspond to the wavelet coefficients. For the point-value multiresolution we have $\widetilde{h}_0=1$ and $\widetilde{h}_n=0$ for $n\neq 0$ and thus $g_1=1$ and $g_n=0$ for $n\neq 1$. The coefficients of h and $\widetilde{g}$ depend on the interpolation order, and in the linear case we have $h_n=\{1/2,1,1/2\}_{n=-1,0,1}$ and $\widetilde{g}_n=\{-1/2,1,-1/2\}_{n=0,1,2}$.

The filter $\widetilde{h}$ illustrates the loose downsampling of the data (simple decimation by a factor of two) when going to coarser scale and it is thus not a low pass filter in the classical sense as no high frequency contributions are eliminated. The corresponding scaling function $\widetilde{\phi}$ is a distribution, and even not in $L^2(\mathbb{R})$. The wavelet ψ is a shifted scaling function and hence not a wavelet with vanishing mean. The corresponding filter g is thus not a band pass filter neither.

These wavelets resulting from Harten's point-value multiresolution are biorthogonal CDF (Cohen–Daubechies–Feauveau) wavelets [12] which are symmetric and compactly supported. For cubic interpolation we have CDF 4/0 wavelets, because the analyzing wavelet has 4 vanishing moments while the synthesizing one has zero vanishing moments. The corresponding filter coefficients h,g and $\widetilde{h},\widetilde{g}$ are summarized in Appendix C (suppl. material) together with an illustration of the functions.

This biorthogonal wavelet decomposition yields good compression properties of the data, as the analyzing wavelet $\widetilde{\psi}$ has vanishing moments. However, there is no separation into scales with well separated frequency bands, as the loosely downsampling does not filter out high frequency contributions (which produces aliasing) and, since the synthesizing wavelet has no vanishing moment, all frequencies are mixing up. For denoising [26] and Coherent Vorticity Simulation [34] such decompositions are only of limited use and for this reason we are applying Sweldens' lifting scheme to construct second generation wavelets. In the next subsection, the CDF 4/0 biorthogonal wavelets are upgraded to CDF 4/4 [64–66] which yield a reasonable scale and frequency separation.

3.2.1 Lifted biorthogonal wavelets

To improve the frequency selectivity of the point-value multiresolution decomposition we apply (primary) lifting to the wavelet ψ and thus construct a wavelet with vanishing moments. To this end we modify the primary wavelet ψ and the dual scaling function $\widetilde{\phi}$ correspondingly, while the secondary wavelet $\widetilde{\psi}$ and the primary scaling function ϕ remain unchanged. For the filters the lifting operations are applied accordingly [64]. More precisely we determine lifting coefficients s_n such that

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

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

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 $\{\widetilde{h}_n;\ n=-2,\cdots,2\} = \{1/8,-1/4,-3/4,-1/4,1/8\}$, 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. The functions ϕ and ψ as well as their dual functions and filter coefficients are illustrated in Appendix C (suppl. material).

3.3 Nonlinear approximation and thresholding

The biorthogonal wavelet representation of the solution v (Eq. (3.3)) can be used for introducing an adaptive grid. To this end we apply thresholding to the detail coefficients $\widetilde{v}_i^J = \langle v, \widetilde{\psi_i^J(x)} \rangle$ and mark details with magnitude below a constant, level-independent threshold C_ε, precisely if $|\widetilde{v}_i^J| / \|v\|_\infty < C_\varepsilon$. As a detail coefficient $\widetilde{v}_i^J$ is directly related to a grid point x_{2i+1}^{J+1}, we could remove the corresponding grid points. In our block-based structure, a block is coarsened only if *all* its details are smaller than C_ε, *i.e.*, some insignificant points can be included. The thresholding error can be estimated in the L^∞ norm and is directly proportional to the threshold, *i.e.*, after N_t time steps we have,

$$\|v_{C_\varepsilon} - v\|_\infty \overset{\mathrm{def}}{=} e_{\mathrm{MR}} \leq N_t C_\varepsilon, \tag{3.8}$$

as shown in [13, 59]. This upper bound assumes a worst-case scenario that the filtering introduces an error $\mathcal{O}(C_\varepsilon)$ in all N_t time steps, which of course depends on the problem. For example in pure transport, the production of fine scales is related only to the lack of translation invariance of the multiresolution transform. Eventually, if no velocity is imposed, the error is in fact $e_{\mathrm{MR}} \leq C_\varepsilon$. Thus, whether the number of time steps enters Eq. (3.8) depends on the non-linearity. Assuming the pure transport-based CFL condition, we have $N_t \propto \Delta x^{-1}$.

3.4 Extension to higher dimensions and vectors

Until now, our description of the multiresolution framework has been limited to the scalar 1D case with a generic variable v. We eventually apply it to the 3D ACM equations. Extensions to dimension larger than one are obtained by tensor product of the one-dimensional wavelets. For multiresolution constructions in two dimensions we obtain wavelets in the horizontal, vertical and diagonal directions, while in three dimensions we have seven directions. For details we refer to [50]. For vector-valued quantities, *e.g.*, the state vector $\underline{V} = (\underline{u}, p)$, the biorthogonal wavelet expansion is applied to each component of the vector and thus a vector-valued wavelet series is obtained. Hence the coefficients become vector-valued, but not the basis functions. For thresholding, wavelet coefficients of each vector component are normalized by the L_∞-norm of the corresponding component.

3.5 Grid prediction and time stepping algorithm

The algorithm to advance the numerical solution $\underline{V}(t^n, \underline{x})$ (state vector containing $\underline{u}$ and p) on the block-structured grid $\mathcal{G}^n$ to the new time level t^{n+1} is as follows:

1. Refinement. Given $\underline{V}(t^n, \underline{x})$ on the grid $\mathcal{G}^n$, we first extend the grid by adding neighbors in position and scale. This concept of a safety zone, introduced in [48], ensures that the solution is tracked in scale and space. Adding neighbors in scale captures newly emerging scales due to the quadratic nonlinearity, which can produce scales twice as small in one time step. The grid $\mathcal{G}^n$ is thus extended to $\widetilde{\mathcal{G}}^n$ by refining all blocks by one level.

2. Evolution. On the extended grid $\widetilde{\mathcal{G}}^n$, we solve Eqs. (2.5)-(2.6) using finite differences with explicit time-marching.

3. Coarsening. We compute the detail coefficients of $\underline{V}(t^{n+1}, \underline{x})$ (i.e., all velocity components and pressure) for each block, and tag it for coarsening if the largest detail is smaller than the prescribed threshold C_ε. Next, the tag is removed if coarsening would result in a non-graded grid, and finally the tagged blocks are merged with their sister blocks. This procedure is repeated until no more blocks can be coarsened.

This well established algorithm (see, *e.g.*, [24, 45, 46, 48, 61]) is at the core of our adaptive strategy. In practice, the above algorithm is modified by imposing an upper bound for the refinement, *i.e.*, a level $J_{\max}$ is introduced at which no refinement is allowed anymore.

The largest possible value in the current implementation is $J_{\max}=18$. This is done not only because of finite computational resources, but also in the interest of balancing the discretization- and thresholding error. While the former error depends mainly on Δx (and thus directly on $J_{\max}$), the latter is determined by C_ε. The relation between both is discussed below.

If the maximum level $J_{\max}$ is reached after the refinement stage and the detail coefficients are significant after the evolution step, we can choose between two possibilities: either, the corresponding block is coarsened, even though its details are significant, or it remains at $J_{\max}$. The former option corresponds to dealiasing and avoids accumulation of energy in high wavenumbers, if viscous dissipation is not sufficient to stabilize the simulation. In low Reynolds number simulations, it can be omitted for the benefit of improved precision, while in most simulations it is required for stabilization.

While the volume penalization term does usually result in large detail coefficients near the fluid-solid interface, it can happen that for large values of C_ε the grid is coarsened at the interface. In this case, oscillations appear in the aerodynamic forces, which are undesirable. Therefore, we use the mask function χ as secondary criterion for block coarsening, *i.e.*, a block is not coarsened if it satisfies $\max(\chi)-\min(\chi)>0$. Consequently, blocks containing the fluid-solid interface are coarsened only if the dealiasing option is used. The initial grid at $t=0$ is created in an iterative process, starting from the coarsest level $J_{\min}$ and likewise takes χ into account.

3.6 Balancing modeling, discretization and thresholding errors

The numerical solution of the Navier–Stokes equations with artificial compressibility on adaptive grids and volume penalization involves five parameters in total, ACM (C_0), penalization (C_η), time (Δt), space (Δx) and thresholding (C_ε). The accuracy of the computation with respect to the reference solution, *i.e.*, the exact solution of the incompressible Navier–Stokes equation, can be estimated from the various error contributions, namely, the artificial compressibility (Eq. (2.4)), penalization (Eq. (2.8)), discretization (Eq. (2.9)) and thresholding (Eq. (3.8)) errors and then applying the triangle inequality, we find,

$$\left\| \underline{V}_{\mathrm{ACM},C_\eta}^{\Delta x,\Delta t,C_\varepsilon} - \underline{V}_{\mathrm{INC}} \right\|_2 \le \mathcal{O}(C_0^{-2}) + \mathcal{O}(C_\eta^{1/2}) + \mathcal{O}(\Delta x^4) + \mathcal{O}(\Delta t^4) + \mathcal{O}(C_\varepsilon). \tag{3.9}$$

A suitable choice of the parameters must take their respective scaling into account in order to balance the different errors. In Section 2.3 we also already discussed the balancing of discretization and penalization errors, which formally reduces the convergence in Δx to first order, (Eq. (2.11)). Requiring all errors to decrease simultaneously by the same amount, we find in the case without penalization the following relations

$$C_\varepsilon \propto \Delta x^5, \quad C_0 \propto \Delta x^{-2}, \tag{3.10}$$

while in the penalized case we have

$$C_\varepsilon \propto \Delta x^2, \quad C_0 \propto \Delta x^{-1/2}, \quad C_\eta = \left(K_\eta^2/\nu\right)\Delta x^2, \tag{3.11}$$

where $0.05 \le K_\eta \le 0.4$. In addition, as described above, we fix the sponge constant to $C_{\text{sp}} = L_{\text{sp}}/C_0\tau$ with $\tau = 20$ (see Appendix D in suppl. material for details).

4 Parallel implementation

4.1 Data structure

Previous work on wavelet-based adaptivity exploited the fact that the fast interpolating wavelet transform can be used to assign one wavelet coefficient to each grid point, and hence allows judging for each point if its wavelet coefficient is important or can be discarded [43]. The resulting grids are as sparse as possible, and each point is a node in a tree data structure [59] or hash-table [52]. The difficulty is that these data structures are poorly aligned with the memory access of typical modern CPUs. Moreover it creates administration overhead for each grid point. If the CPU requires a data point for computing, the vicinity of this point in the 1D memory is also transferred to the CPUs cache, which explains the high performance of structured grids. To turn the achievable sparsity into CPU time compression, a block-based hybrid data structure is a suitable choice. In the context of wavelet-based multiresolution, block-based grids have first been proposed by Domingues et al. [24] and later, independently, in [57]. The basic idea is however older; for adaptive computations without wavelet-support, they have been pioneered by Berger and Collela [5] and further developed by Deiterding and coworkers [15–17]. An excellent overview over recently developed software packages can be found in [62].

Block-based grids are locally structured and for the relations between blocks, a flexible data structure has to be used. In our implementation, we use a tree-based structure for this task, and define blocks of size B_s^D. All blocks have the same number of points, which is the same in all directions in the present computations. The number of entries in the tree data structure, and thus its overhead, is reduced by the same factor B_s^D. On the other hand, the total number of grid points N_{points} is increased, until for (very) large B_s, an equidistant monoblock grid is recovered. Fig. 1 illustrates the concept of our data structure. The grid layout contains blocks of the same size on different levels, while the memory layout is a simple, contiguous array of size $B_s \times B_s \times B_s \times N_b$. Since the number of blocks N_b changes during the simulation, our code allocates an array with the largest possible number of blocks for the given memory, which is specified in the program call using the flag `--memory=40.0GB`. Memory allocation is done once during the initialization and is not deallocated again until the termination of the code. This type of memory management is suitable for high-performance supercomputers, where typically only one application is running on a given processor.

Since the number of blocks N_b in the tree data structure is typically low ($N_b = \mathcal{O}(10^5)$ blocks, while $N_{\text{points}} = N_b B_s^D = \mathcal{O}(10^{10})$ in 3D applications) there is no need for an external high-performance tree library, such as `p4est` [8]. Here, we design a simple tree library based on the work of Gargantini [35, 36], where octal numbers are used to encode the position of an element in the tree. From the octal number, level and position of the block

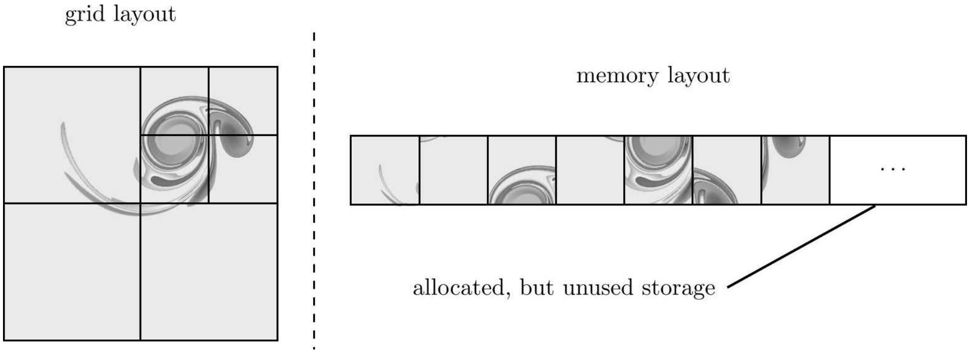


Figure 1: Data structure. Left: grid layout, composed of N_b=7 blocks of the same size but at different positions and levels. Right: memory layout, the grid is stored as simple, contiguous array. Memory allocation done once on startup, but only the portion required at time t is used.

can be computed, as well as the IDs of neighboring blocks. It constitutes, along with a refinement flag and level indicator, the metadata, which is separated from the actual block data named heavy data. The metadata is stored redundantly on all CPUs and is kept synchronized, so that neighbor lookup is a local process, while the heavy data is process-local.

4.2 Parallelization aspects

Parallelization appears in our implementation with two functions: to ensure that each MPI rank has the same amount of blocks to compute (load balancing), and to synchronize the layer of ghost nodes.

The creation and removal of blocks during the simulation can lead to an imbalanced distribution of blocks among processes. As we use blocks of identical size, load balancing is guaranteed if the number of blocks on each process is the same. The load balancing procedure first computes the index of all blocks on a space-filling Hilbert-curve [42, 69], then distributes contiguous chunks of blocks to the processes. Special care is devoted to keeping the required number of transfers as small as possible. The Hilbert-curve preserves the locality of blocks and minimizes the interface between processes, as shown in Fig. 2. We balance the load at the end of every time step after grid coarsening. As in our algorithm all blocks are refined prior to making a time step, refinement increases N_b exactly 2^D-fold but keeps load balancing intact.

Any parallelization by decomposition of the domain requires communication between the processes, either in the form of flux transfer or ghost node synchronization. Our blocks are equipped with a layer of n_g ghost nodes (where n_g depends on the support of the interpolation and discretization stencils) that overlaps with adjacent blocks. During the synchronization of the ghost nodes, the processes first prepare the required data, *i.e.*, apply restriction and prediction operators if required, and copy them into a buffer, which is then sent to the receiving process. The receiver then extracts the data to

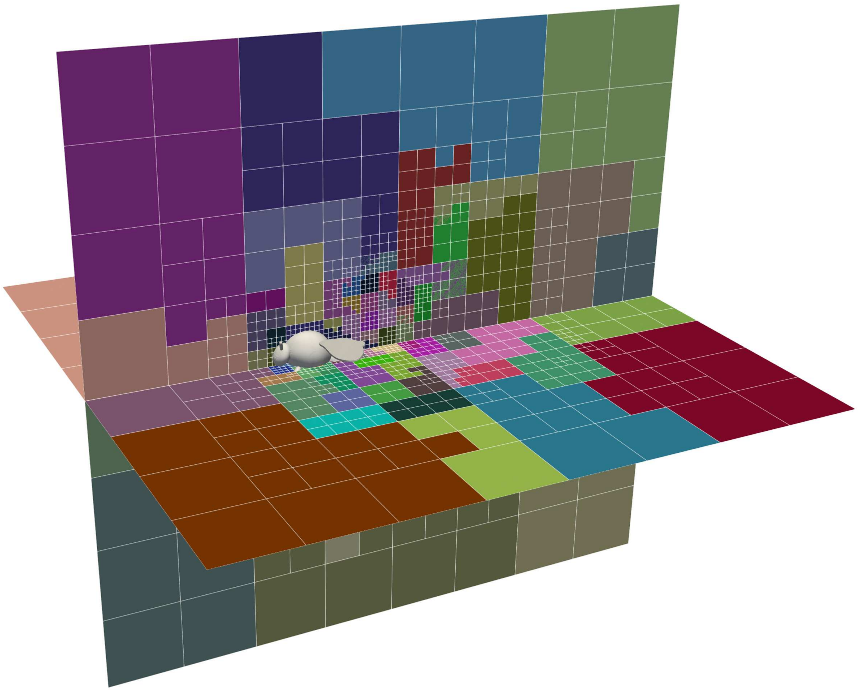


Figure 2: Parallel load balancing. Shown is a snapshot of the (fine resolution) bumblebee simulations presented in Section 5.3. In the two planes, the block borders are shown along with the `MPI` rank. Here, $N_{\text{CPU}} = 960$, and each `CPU` is assigned a random color. The space-filling curve results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

the ghost node layers of each block. We skip more details on this technical part in the interest of brevity. The handling of block-block interfaces *via* the ghost nodes is linked to the precise definition of blocks and grid, and it is critical for numerical stability. We elaborate on our choices in Appendix B (suppl. material).

Parallelization of an adaptive code is inherently more complicated than for a code using static grids. The number of blocks can vary significantly during a simulation and choosing the same number of processes throughout the simulation is often not an efficient choice. On most supercomputers, a simulation is interrupted after a given time limit is exceeded, which is usually of the order of 24h. We exploit this fact to decide upon resubmission of a simulation if the number of CPUs is increased, decreased or can remain unchanged. We will discuss the performance in greater detail in Section 5.4.

5 Numerical results and performance

All simulations presented in this paper can be reproduced using the code[†] and the parameter files from the supplementary material (See Appendix F for an overview).

[†] https://github.com/adaptive-cfd/WABBIT

5.1 Three vortices

As a quantitative validation test for the fluid part in two-dimensions, we consider the vortex merger proposed in [44]. The setup involves no walls and is a good starting point for the validation of our new numerical method. At time $t=0$, three Gaussian vortices (Fig. 3A,B) are set into the periodic domain of size $L=2\pi$, two with positive ($\Gamma=1$) and one with negative ($\Gamma=-1/2$) circulation. The initial vorticity field is given by

$$\omega(x,t=0)=\sum_{i=1}^{3}\frac{\Gamma_i}{\pi\cdot a^2}\exp\left(-((x-x_{0,i})^2+(y-y_{0,i})^2)/a^2\right),$$

where $a=1/\pi$ and the vortex centers $(x_{0,i},y_{0,i})$ are located at $(0.75,1)\pi$, $(1.25,1)\pi$ and $(1.25,1.25)\pi$. Note that initially, the mean vorticity is small but not zero. The vortices then interact nonlinearly, but their evolution remains deterministic. Due to the low viscosity of $\nu=5\cdot10^{-5}$, thin vortex filaments develop during the merging of the two positive vortices (Fig. 3A). We stop the simulation at $t=20$ (Fig. 3C), and use this time instant for comparing different simulations. The creation of small scales due to the nonlinearity is an important test for our adaptive framework.

We perform four types of numerical simulations for this problem:

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.

Those simulations allow us to assess (i) the model error of the ACM, (ii) the discretization error and (iii) the thresholding error. Combined, they allow us to confirm the relation between the three numerical parameters of the problem (C_o, Δx, C_ε) presented in Eq. (3.10). Instead of Δx, it is however more practical to use the maximum refinement $J_{\max}$, which is related to Δx via $\Delta x=2^{-J_{\max}}L/(B_s-1)$. In all ACM simulations of this problem, we set the pressure damping $C_\gamma=1$, as the sponge technique cannot be used for periodic domains.

(i) Compressibility error. We first consider the Fourier pseudo-spectral solution of the artificial compressibility Eqs. (2.5)-(2.6). Comparing the spectral ACM simulations with the incompressible reference solution from [44] allows us to evaluate the compressibility error alone. We compute the reference solution with spatial resolution of 1536×1536 modes using the `FLUSI` code [33], which uses the same Fourier pseudospectral scheme

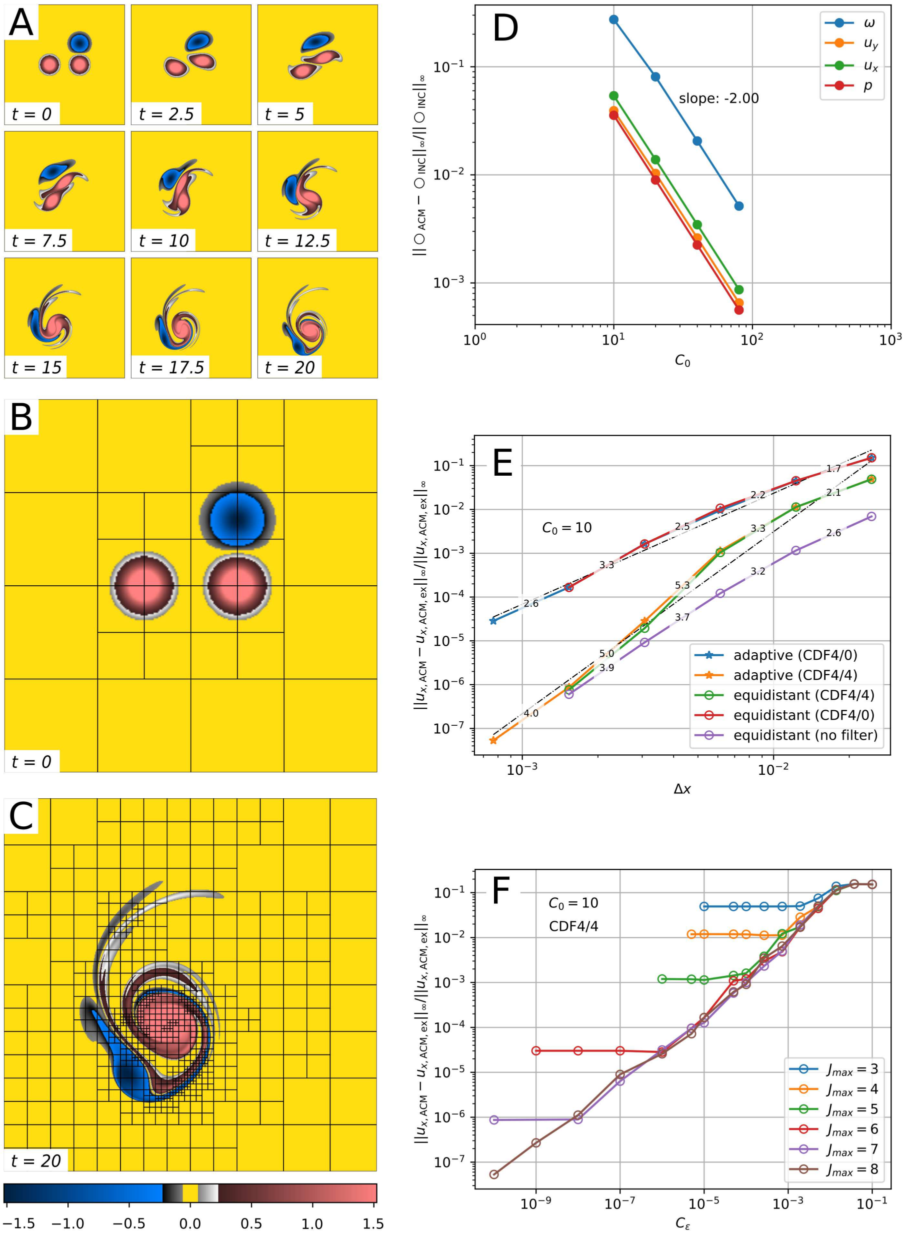


Figure 3: Three vortices simulations. A-C: Time evolution of the vorticity field in a simulation with $B_s = 33$, $C_\varepsilon = 10^{-5}$, $J_{\text{max}} = 8$ and $C_0 = 10$ using the CDF 4/4 wavelet. Time evolution (A), initial condition (B) and final time used for error calculation (C). D: Decay of the compressibility error with respect to the incompressible solution as a function of C_0, computed using a non-adaptive spectral discretization. Second order is observed. E: Decay of the error w.r.t. the spectral ACM solution as a function of Δx for $C_0 = 10$. Shown are equidistant computations with and without wavelet filtering and adaptive simulations with C_ε small enough for the discretization error to be dominant. The adaptive computations preserve the accuracy of the equidistant ones. F: Decay of the error using CDF 4/4 w.r.t. the spectral ACM solution as a function of C_ε for different J_{max} and $C_0 = 10$. Leftmost points of each line are also shown in E.

as in [44]. Time integration is done with the classical 4th order Runge–Kutta scheme and $\mathrm{CFL} = 0.15$. For all times $t \geq 0$, we verified that the isotropic Fourier spectrum decays exponentially for large wavenumber k below machine precision, hence both the reference solution and the spectral ACM simulations can be considered highly accurate. Fig. 3D shows the resulting convergence in C_0, confirming the second order convergence from Eq. (2.4).

(ii) Discretization error. We next evaluate the discretization error of the finite difference scheme on equidistant grids, fixing $C_0 = 10$ for computational convenience, and compare with the corresponding spectral solution of the ACM equations. The equidistant computations with resolutions from $N = 256$ to 4096 are performed using the `WABBIT` code with adaptivity turned off. In this test case, the viscosity is sufficiently elevated to stabilize the simulation without filtering at the maximum level J_{max}. Therefore we perform three types of equidistant computations shown in Fig. 3E, CDF 4/0 (red line), CDF 4/4 (green) and without (purple) a filter. Without filtering, the method is clearly fourth order accurate. If wavelet filtering at J_{max} is used, which corresponds to removing the detail coefficients, the error in the CDF 4/4 solution increases slightly, as expected, compared to the case without filtering. Note that filtering is usually required for stabilization. The convergence rate locally varies, but a linear least squares fit to the data yields a slope of 4.04. With the CDF 4/4 wavelet, we can hence preserve the accuracy. The CDF 4/0 method on the other hand increases the error and deteriorates the convergence rate. We anticipate that this is due to the lack of scale separation in the CDF 4/0 wavelet.

(iii) Thresholding error. We now consider fully adaptive simulations, where we fix $C_0 = 10$ and use the refine–evolve–coarsen cycle in each time step to remove aliasing errors. The block size B_s is mainly a parameter that impacts the computational efficiency, which is crucial only for 3D simulations. We here fix $B_s = 33$ for all 2D simulations and discuss the performance as a function of B_s later in Section 5.4. Two snapshots of an adaptive computation are shown in Fig. 3B,C. Visibly, the initial grid contains less blocks than the terminal one. The refinement is concentrated, as expected, in regions with sharp gradients.

In adaptive simulations, the thresholding error arises in addition to discretization and compressibility errors. We compare with the spectral ACM solution for C_0 fixed, hence the compressibility error is absent. Fig. 3F shows the error as a function of C_ε for different J_{max}. For large C_ε, all simulations run on the coarsest possible level ($J_{\mathrm{max}} = 1$) and therefore yield the same error. When C_ε is decreased, the error generally decreases, until a saturation at a level-dependent value of $C_{\varepsilon,\mathrm{opt}}(J_{\mathrm{max}})$ is observed. At this point, the discretization- and thresholding error are of the same size. Decreasing C_ε further can thus not increase the precision, as the discretization error then dominates. Before the saturation, the error decays $\propto C_\varepsilon$. Fig. 3E shows the convergence as a function of Δx when using CDF 4/0 (blue line) and CDF 4/4 (orange), where we use the smallest C_ε used for each $J_{\max}$. The obtained curves overlap the equidistant one almost perfectly – we can hence conclude that we preserve the accuracy and the order of the discretization scheme on adaptive grids.

5.2 A flapping wing

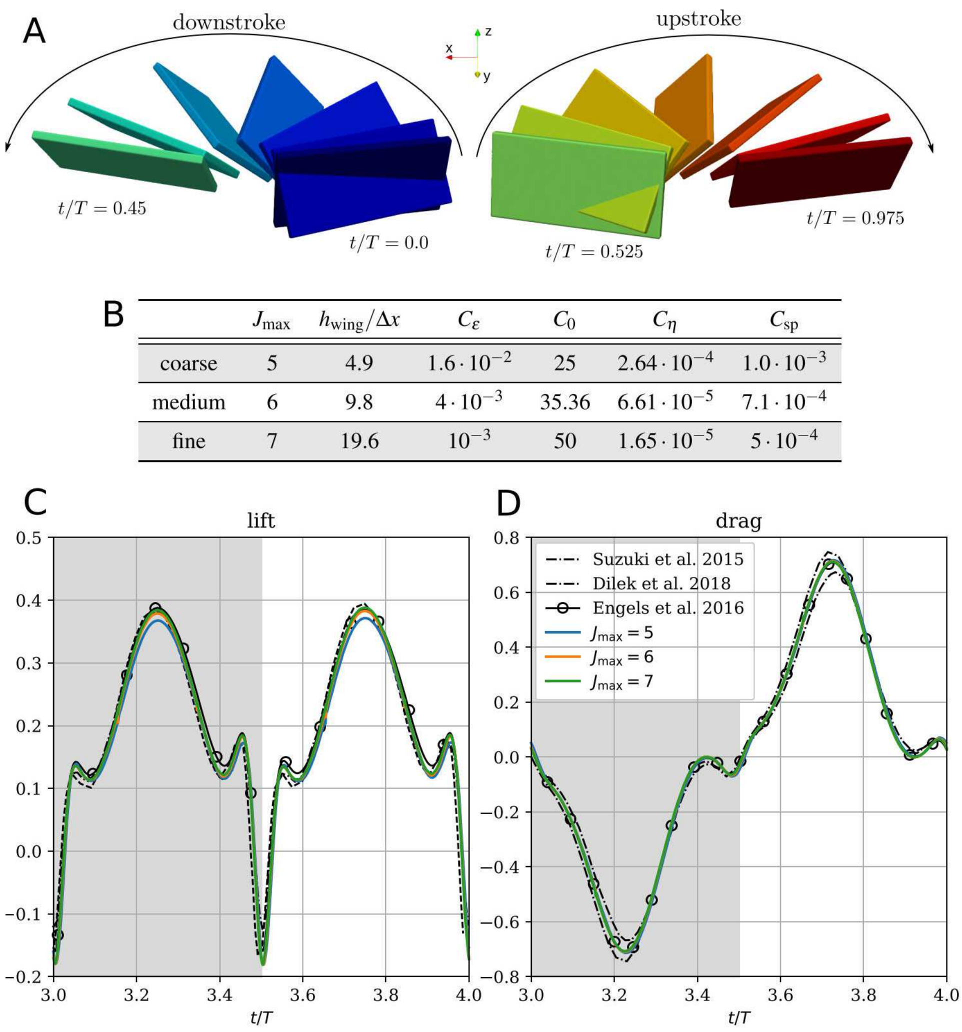
As example of a time-dependent geometry in 3D, we consider the test case proposed by Suzuki et al. [63], Appendix B2. It has also been considered as validation case in our previous work [33] and by Dilek et al. [22]. We therefore limit the description of the setup to a minimum and refer the reader to [33] for details of the geometry and wingbeat kinematics. The test consists of a single, rectangular, rigid flapping wing with finite thickness $h_{\text{wing}}=0.04171$ and length $R=1$. No inflow is imposed; the setup mimics hovering flight. The wing moves in an horizontal stroke plane, its prescribed wingbeat motion is visualized in Fig. 4A. The two half cycles, conventionally termed up- and downstroke, are symmetrical. The size of the computational domain is chosen large enough to be a good approximation for an unbounded flow, in our case we set $6R\times 6R\times 6R$, which is slightly larger than [22, 63]. At the outer border, we impose homogeneous Dirichlet conditions on $\underline{u}_{\infty}=0$ and $p_{\infty}=0$ using the sponge term in Eqs. (2.5)-(2.6). The initial condition is $\underline{u}(t=0)=p(t=0)=0$.

As penalization is used for the wing, the scaling relations (3.11) are used, and we use the constant $K_{\eta}=0.365$ justified from our previous work [33]. Fig. 4B summarizes the parameters used for a coarse, medium and fine resolution simulation.

Because the CDF 4/0 wavelet has a smaller support (two ghost nodes compared to six for the CDF 4/4), it is for a given grid less expensive to compute. For cases with impulsive start, like the ones presented below, we observed artifacts resulting from the strong initial pressure wave when using CDF 4/4 wavelets, which do not appear with the CDF 4/0 ones. We therefore perform all 3D simulations with CDF 4/0. The impact of the wavelet on the solution will be the focus of forthcoming work.

The low Reynolds number of $Re=u_{\text{tip}}c_m/\nu=100$, which is comparable to that of a fruit fly, results in a smooth flow topology. Here, u_{tip} is the mean wingtip velocity and c_m the chord length. The flow displays the characteristic features of flapping flight, namely a strong leading edge vortex and a wingtip vortex. The computational grid shows the expected refinement near the wing, while the resolution in the remaining parts of the domain is dictated by gradedness. Because of the absence of fine-scale structures in the flow, the grid is in practice determined uniquely by the mask function χ, and the parameter C_{ε} is not important. Nonetheless, C_{ε} has been scaled according to Eq. (3.11) for consistency. We set a block size of $B_s=23$ (see Section 5.4) and because N_b does not strongly vary over time, the number of CPU is constant throughout the simulation.

Fig. 4C-D compares the lift- and drag force obtained by present simulations with the results obtained by Suzuki et al. [63], Dilek et al [22] and our own previous work [33]. As already noted by Suzuki in their initial publication using two different solvers, some discrepancy between the reference solutions is observed. The difference, integrated over



	$J_{\max}$	$h_{\text{wing}}/\Delta x$	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	$1.65\cdot 10^{-5}$	$5\cdot 10^{-4}$

Figure 4: Suzuki's test case. A: visualization of the wingbeat. The rigid wing moves in an horizontal stroke plane with varying angle of attack, the motion is symmetrical in down- and upstroke. B: computational parameters used in the simulations. C-D: Time evolution of lift and drag and comparison with results from the literature. Results are obtained with CDF 4/0 wavelets.

the stroke period, is of the order of 5%, which is why we do not present convergence plots.

5.3 Bumblebee in forward flight

The last example in our hierarchy of validation cases is the simulation of a model bumblebee. This model has also been used in our previous work [29,30] and it was the first to be studied in fully developed turbulence. In the interest of brevity, we limit the description of the model to a minimum and refer to [29, 30] for further details. The electronic supplementary material to this article contains the parameter files used in the simulation for reproducibility. The penalization method allows for the inclusion of obstacles of arbitrary complexity, and we hence include the bumblebees body. The wingbeat is illustrated in Fig. 5A. We are considering forward flight, which is why the stroke plane is inclined with respect to horizontal. The wing kinematics differs significantly between down- and upstroke. During the downstroke, also called power-stroke, most of the aerodynamic forces are generated, as the wing moves against the flow direction at elevated angle of attack. The upstroke at much smaller angle of attack primarily serves to enable the next downstroke. The wings are rigid and have a thickness $h_{\text{wing}} = 0.025R$. Our bumblebee has a Reynolds number of $Re = u_{\text{tip}} c_m / \nu = 2000$ and as such produces a more complicated flow topology than the previous test [28–30]. Wing motion is prescribed and not the result of an interaction with the fluid. Such coupled simulations will be the focus of forthcoming work.

The domain size is $L=8$, the block size is $B_s=23$ (see Section 5.4) and the other numerical parameters of three simulations levels are given in Fig. 5B. All results are obtained with the CDF 4/0 wavelets.

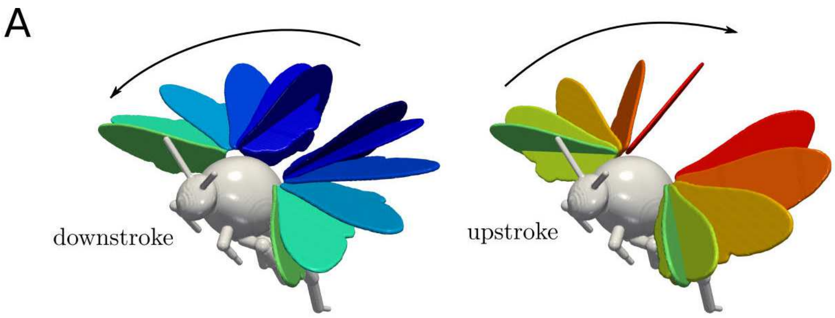
The drag (Fig. 5C) and lift (Fig. 5D) show good convergence and display features typical for flapping wings. The leading edge vortex (LEV, cf. Fig. 6A) is built up during the downstroke and reaches its maximum intensity during the mid-downstroke ($t=2.25$), which results in a peak in the lift force. At the turning point ($t=2.5$) between down- and upstroke, the rapid wing rotation causes large peaks in the forces.

Fig. 6A-C shows the same snapshot of the flow field at $t/T = 2.3$ in all three simulations. Shown is the $|\underline{\omega}| = 50$ isosurface of vorticity magnitude. Visually, the three levels yield similar results for the large scale structures of the flow field. The grids concentrate, as expected, high resolution near the insect, but contrarily to the previous testcase parts of the wake are now likewise are refined to $J_{\max}$, because they are detected by the wavelet thresholding. In the coarse simulation, 3% of the full grid is used, which decreases to 1.3% and 0.6% for the medium and fine resolution, respectively (Fig. 5E).

5.4 Performance

For a specific simulation, three factors are important for the computational performance: the block size B_s, the number of blocks per CPU N_b/N_{CPU} and the number of CPU itself. B_s balances grid sparsity and data regularity and influences the solution, while the other parameters are purely computational. We clarify the influence of these parameters in this section.

The impact of B_s can be divided into to contributions: the grid sparsity that depends on the physics of a simulation and the computational efficiency resulting from the data regularity. The latter depends only on the machine. We therefore study this part as follows. For each value of B_s, a random grid of given N_b is generated, then 15 time steps of the ACM equations are performed on this grid. Each step consists of the refine-evolve-



B

	$J_{\max}$	$h_{\text{wing}}/\Delta x$	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	$3.41\cdot 10^{-5}$	$5\cdot 10^{-4}$

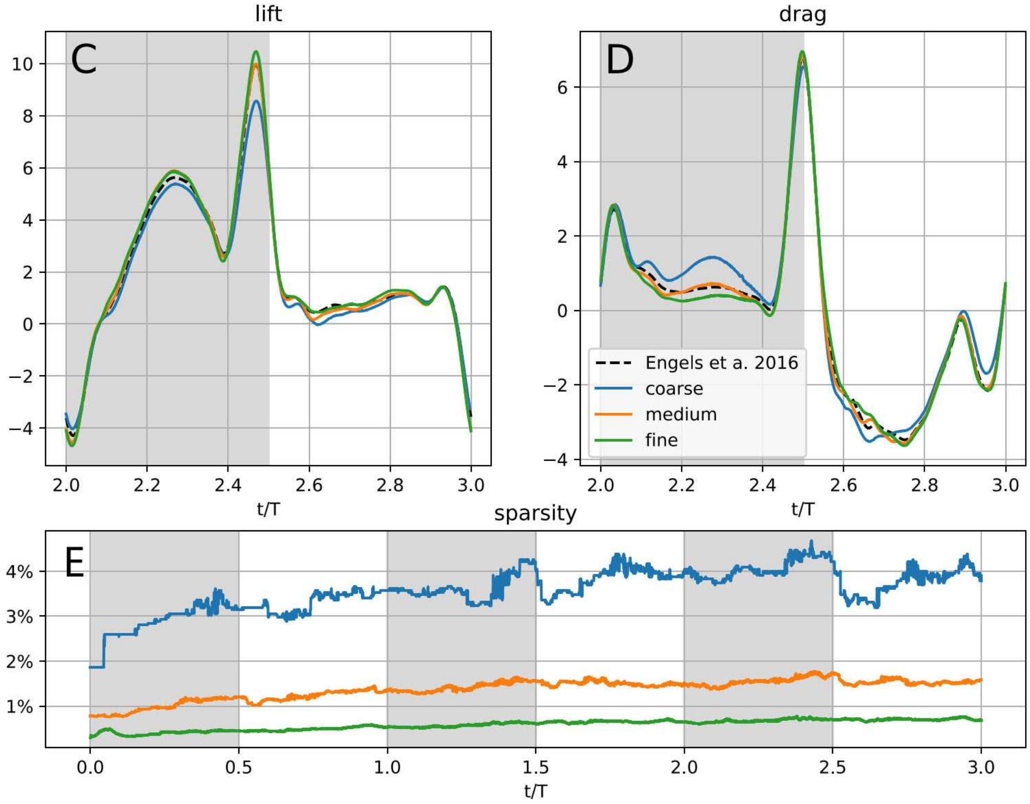


Figure 5: Bumblebee simulations. A: Wingbeat of the bumblebee model in forward flight. During the downstroke, the angle of attack is more elevated than during the upstroke. The stroke plane is inclined to generate a forward force. B: computational parameters for the three levels of simulation. C-D: Lift- (=vertical) and drag (=horizontal) force. E: grid sparsity $N_b/2^{3J_{\max}}$ as a function of time. Results are obtained with CDF 4/0 wavelets.

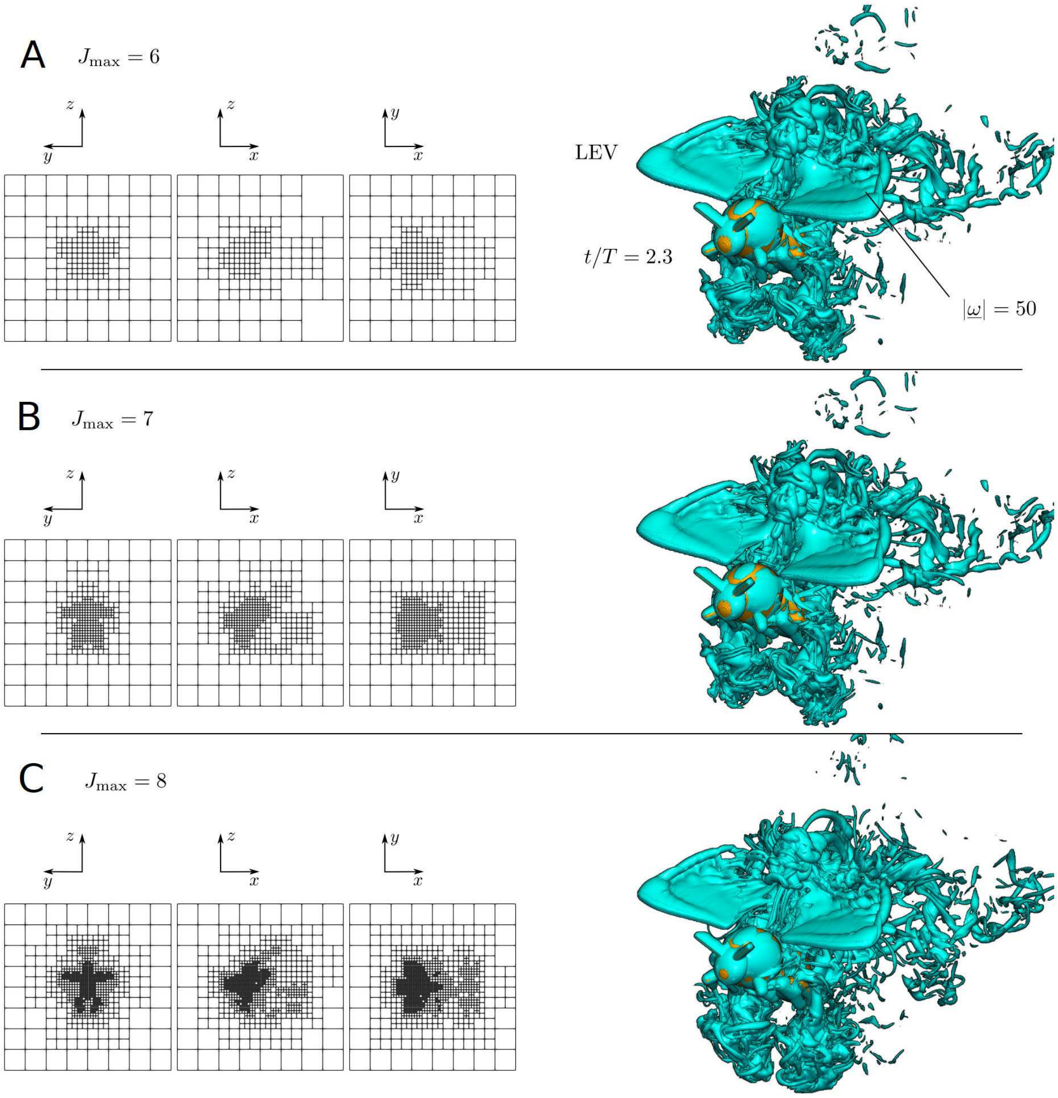


Figure 6: Bumblebee simulations. A (coarse), B (medium), C (fine resolution): Visualization of the flow field ($||\underline{\omega}|| = 50$ isosurface of vorticity magnitude) at $t/T = 2.3$ (right) and 2D projections of the computational grid used at this time (left) for the three simulations. Results are obtained with CDF 4/0 wavelets.

coarsening cycle explained in Section 3.5. As we use random data, the adaptation is set to coarsen everywhere instead of using wavelet thresholding. After measuring the performance on this grid, a new, slightly denser grid is generated. The procedure is repeated until the allocated memory is exhausted.

Fig. 7A shows the CPU time (in CPU seconds) consumed per grid point and per right hand side evaluation, as the RK4 scheme has $s = 4$ stages. This number is hence indepen-

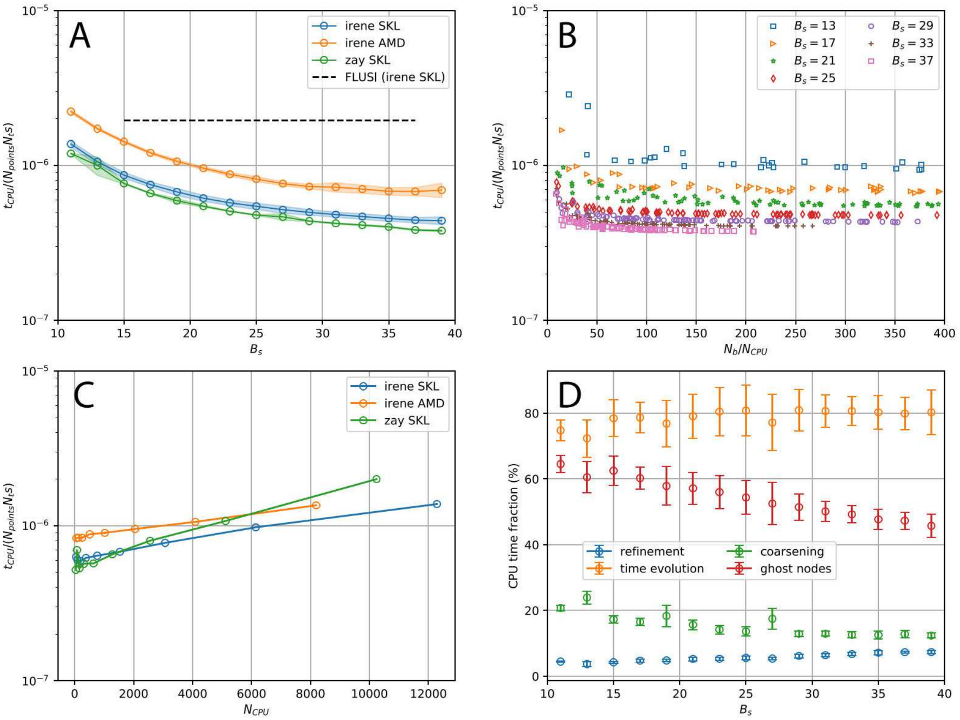


Figure 7: Computational performance on randomly generated 3D grids. A: CPU cost per grid point per right hand side evaluation as a function of B_s, measured on three supercomputers (`Irene SKL`, `Irene AMD`, `Zay SKL`). For comparison, cost using the pseudospectral `FLUSI` code on `Irene SKL` is included as well. Note only one simulation is shown, independent of B_s. Shaded areas correspond to mean ± 1 s.d. B: Cost as a function of N_b/N_{CPU} for different B_s, computed on $N_{\text{CPU}} = 200$ 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 $N_{\text{CPU}} = 200$ cores on `Zay SKL`. Results are obtained with CDF 4/0 wavelets and RK4 scheme ($s = 4$).

dent of the chosen time stepper, as our adaptivity only concerns spatial discretization. We used three different supercomputers, `Irene SKL`, `Irene AMD`[‡], `Zay SKL`[§]. For smaller B_s, the cost increases because the overhead due to adaptivity becomes larger. Larger B_s reduces the cost per point because of the increased data regularity. It is important to compare the performance to other existing codes, therefore the computational cost for the spectral code `FLUSI` [33] is also shown. We choose this code because we can simulate exactly the same bumblebee setup with a completely different approach. In this simulation, the resolution is 2048^3 grid points and $N_{\text{CPU}} = 1152$. The comparably low N_{CPU} ensures best possible performance for the `FLUSI` code. The performance of `WABBIT` is about four times better than that of `FLUSI` for larger B_s, which demonstrates that the additional overhead due to adaptivity is successfully mitigated by the block based data

---

[‡] http://www-hpc.cea.fr/fr/complexe/tgcc.htm

[§] http://www.idris.fr/

structure. Because of the full grid in the `FLUSI` case, the total cost is much more elevated, but on the other hand it relies on highly optimized `FFT` libraries. The similar performance per grid point and time step is thus a strong indicator for competitive performance of the `WABBIT` code.

Using dynamical grids the work load per CPU is not constant. It is therefore important to verify that the performance remains indeed constant over a large range of blocks per CPU, N_b/N_{CPU}, as shown in Fig. 7B, computed using $N_{\mathrm{CPU}}=200$ on `Zay SKL`. For $N_b/N_{\mathrm{CPU}}>50$, the performance is nearly constant, while it drops for smaller values.

With increasing N_{CPU}, reasonable values for N_b/N_{CPU} are obtained only for very large simulations. At this point, the overhead of the global topology management can eventually become important and increases the cost per grid point (Fig. 7C). Improving this scaling behaviour is left for future work.

Fig. 7D shows the fraction of CPU time spent on refinement, evolution and coarsening, which sum to 100%. They remain approximately constant over the range of B_s, while the ghost node synchronization and hence the parallelization overhead are much larger at small B_s. Synchronization is performed in all three of refinement, evolution and coarsening. Approximately 70% –80% are spent on computing the actual right hand side. We should note at this point that the ACM equations considered here are extremely simple and can be solved very efficiently. For more complicated equations, such as the full compressible Navier–Stokes equation, the portion of time spent on time evolution can easily exceed 90%.

For the simulations presented earlier, we have not yet discussed how the block size B_s=23 has been chosen. Prior to performing the actual simulations, we performed a series of runs with varying B_s on the bumblebee configuration. As the smallest Δx depends on B_s, we cannot simply change B_s without altering results. Therefore, we took advantage of the fact that for an exterior flow, the domain size L is an artifact from the simulation rather than a physical quantity ($L\to\infty$ in reality). Thus, for every B_s we modified L to obtain the same Δx. All simulations run until $t=0.15$, and the other parameters are $J_{\max}=8$, $C_\varepsilon=10^{-3}$, $C_0=20$ and $C_\eta=1.65\cdot 10^{-4}$.

Fig. 8A shows the total number of points on the grid, averaged over the time span of the simulations. It is observed that as the computational cost per point decreases (Fig. 7A), the total number of points increases. Both influences combined yield Fig. 8B, which shows the total cost per simulation time unit. It presents a minimum for an intermediate value of B_s, where a compromise between sparsity and data regularity is achieved. We therefore choose $B_s=23$, but also note that any value between 15 and 30 would have resulted in near-optimal performance

We conclude the discussion on performance with a strong scaling test for the bumblebee simulations. We used the parameters from the 'fine' case (Fig. 5B). Identical simulations run until $t=0.1T$ on increasing N_{CPU}. During this time span, $N_b\in[49344,\ 84176]$. Reasonable but not perfect scaling is observed for all tested supercomputers. For the tests with largest N_{CPU}, $\min(N_b/N_{\mathrm{CPU}})$ is 14, 8 and 13 on `Irene SKL`, `Irene AMD` and `Zay SKL`, respectively.

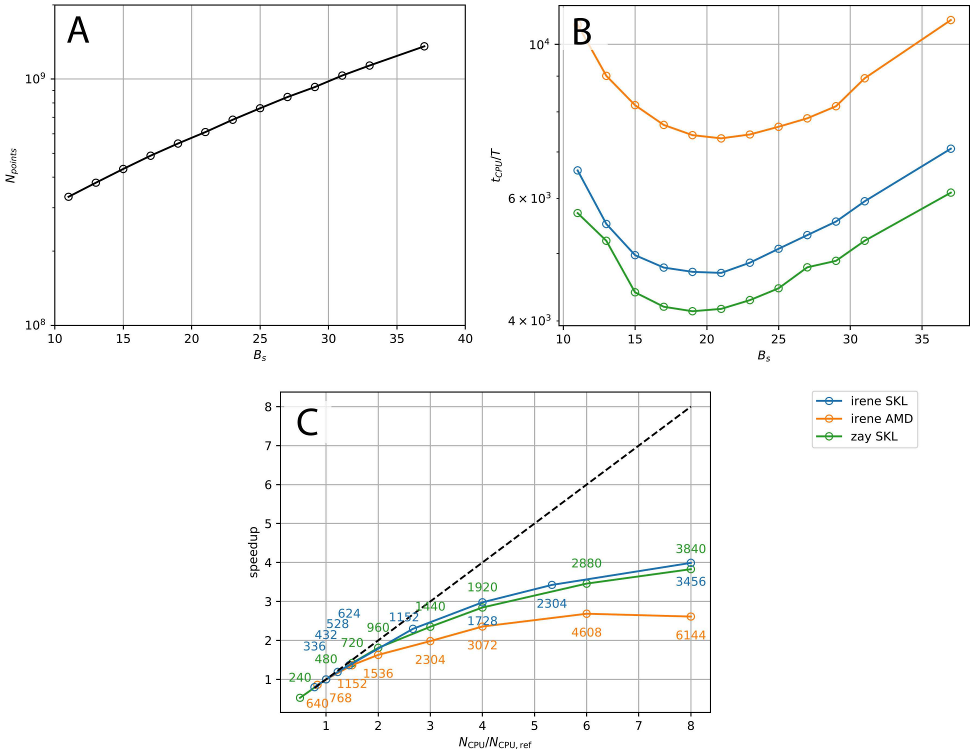


Figure 8: Computational performance in bumblebee simulations. A: time-averaged total number of points on the grid $N_{\text{points}} = N_b B_s^3$ as a function of B_s B: Total cost per time unit of a simulation with different B_s on different supercomputers. C: Strong scaling test on the 'fine' (cf Fig. 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.

In summary, a good performance of the `WABBIT` code is observed, as long as the load per processor, N_b/N_{CPU}, is sufficiently large. The specific value is machine-dependent and between 25 and 50 for the tested machines.

6 Outlook: Bumblebee in the wake of a fractal tree

To give an outlook and to illustrate the flexibility of the present adaptive wavelet approach to compute flows with multiple scales, we simulate the flow of the bumblebee model, described previously, in the wake of a model flower, corresponding to a bio-inspired turbulence generator. The latter is composed of an array of rigid cylinders arranged in a fractal manner, largely motivated by plants. For a detailed description of the geometry we refer the reader to [27]. Both bumblebee and fractal tree are simulated using the volume penalization method in a large cubic domain of size $L = 64R$, where R is the wing length of the insect. Fig. 9 (top, left) illustrates the setup. The thinnest cylinders in the fractal tree have a diameter of $d_{\min} = 0.114R$ and thus we have a Reynolds number of

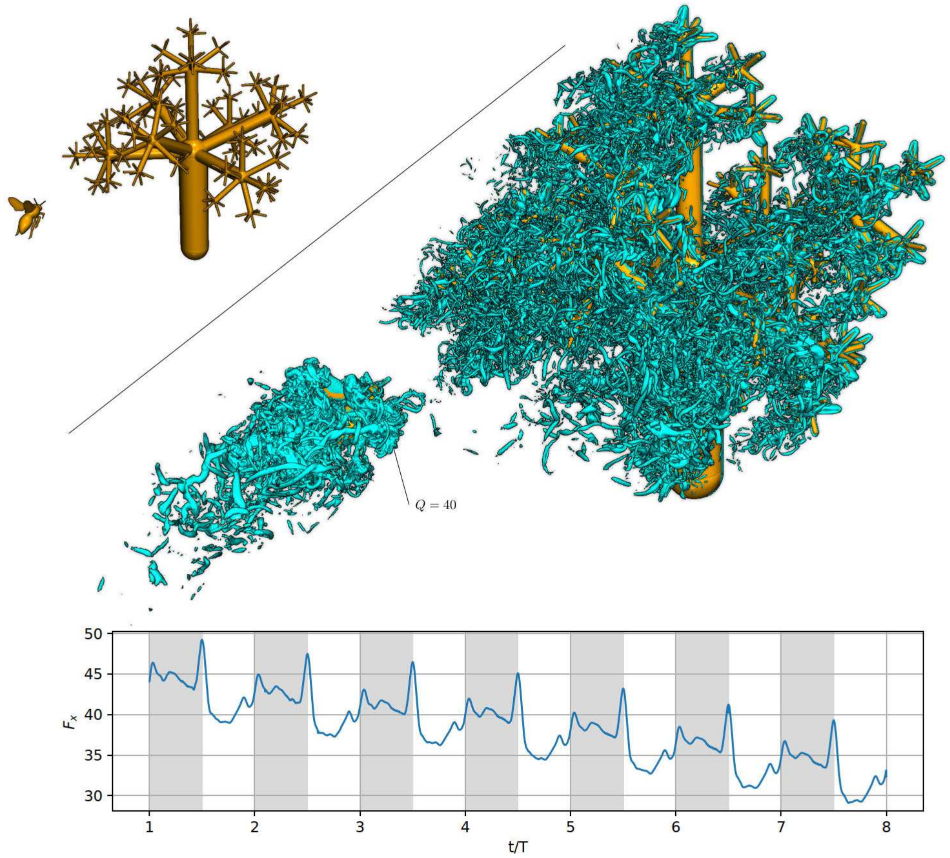


Figure 9: Bumblebee behind a fractal tree. Top part shows the setup consisting of a bumblebee and a tree-inspired fractal turbulence generator composed of rigid cylinders. Right part illustrates the flow field with an isosurface of the Q-criterion. Time in figure is $t/T = 8.0$, results obtained with 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.

$\mathrm{Re} = u_\infty d_{\mathrm{min}}/\nu = 250$. The other numerical parameters are chosen as in the coarse simulations of the bumblebee alone, described in Section 5.3. The simulation is performed using CDF 4/0 wavelets where $C_\varepsilon = 4\cdot 10^{-2}$. The flow field is visualized in Fig. 9 with an isosurface of the Q-criterion. It shows the turbulent wake produced by the tree model and the wake of the bumblebee after eight wingbeats. The maximum number of refinement levels is $J_{\mathrm{max}} = 9$ and the grid is composed, on average, of $N_b = 1.8\cdot 10^5$ blocks with a total number of $2.2\cdot 10^9$ grid points. The dynamically adapted grid allows us to use a large domain size, which would result in 11776^3 points on the corresponding, uniform grid with the same Δx. We hence use on average only 0.15% of the uniform grid. Note however that the domain size could possibly be reduced. A detailed study on the flow physics of the bumblebee-plant interaction will be reported elsewhere.

7 Conclusions

This article presents a novel framework for wavelet-based adaptive simulations of turbulent flows with a wide range of dynamically active scales in complex, time-dependent geometries on massively parallel computers, with a specific focus on flapping insect flight. It is implemented in the `WABBIT`-code, which is open-source in order to maximize its accessibility and its utility to the scientific community. We base our physical model on the original combination of artificial compressibility and volume penalization, and discretize the governing equations using classical centered finite differences on a locally uniform Cartesian grid together with a Runge–Kutta scheme for time integration, both of fourth order. Motivated by the intermittent nature of our applications, we employ biorthogonal interpolating wavelets to introduce a dynamically adaptive grid tracking the solution in space and scale. In contrast to many previous wavelet-based implementations, the data structure is based on locally uniform blocks, which leads to an improved performance on modern CPUs. Thresholding the wavelet coefficients allows us to decide if a block can be coarsened at the end of a time step. As a block contains multiple wavelet coefficients but can only be coarsened as a whole, the number of points per block is a numerical parameter balancing the influence of data compression and CPU efficiency. The global grid topology is encoded using a custom tree-like data structure, as is natural in the wavelet-context. Even though our method has excellent compression properties, the applications we have in mind require many blocks and thus using massively parallel supercomputers is mandatory. To this end, blocks are distributed among MPI processes and the layer of overlapping ghost nodes is synchronized. This implementation yields high parallel efficiency using thousands of CPUs while maintaining both memory- and CPU time compression.

Through numerical estimations and a series of validation cases, we discussed how the parameters for this model have to be chosen in order to simultaneously reduce all occurring errors, *i.e.* compressibility, penalization, discretization and thresholding errors, while maintaining computational efficiency. We also assessed the impact of the block size on the performance and illustrated how it is chosen to yield maximum efficiency. Considering different supercomputers we furthermore showed that the performance per grid point and time step is very similar to that of the previously developed Fourier pseudospectral code `FLUSI`. Because this spectral code relies on the fast Fourier transform, a task for which highly tuned libraries are used, we can conclude that our new framework offers excellent performance and can indeed translate the sparsity of the grid into CPU time savings.

If we consider, for instance, the case of the high-resolution bumblebee computation with corresponding uniform grid resolution 5632^3, we showed that the adaptive wavelet-based computation is, for similar accuracy, about 160 times faster than the classical Fourier pseudospectral code. While this is an optimistic scenario because the domain size can still be reduced, it still illustrates the strength of our novel approach.

We finally presented a simulation of an insect together with a fractal tree, which will be used in future work to study interactions of insects with their environment. Such multiscale simulations are out of reach for other classical numerical approaches, in particular spectral codes, and we are not aware of other codes suitable for such problems.

In future work we are planning to perform adaptive computations of insects using realistic bodies from micro CT data of real insects and mass-spring models for flexible wings [68] to perform adaptive fluid-structure interaction simulations.

Acknowledgments

TE, KS, MF gratefully acknowledge financial support from ANR (Agence Nationale de la Recherche), grant n^o 15-CE40-0019, and DFG (Deutsche Forschungsgemeinschaft), grant n^o SE 824/26-1, project AIFIT (Aerodynamics of Insect Flight In Turbulence). The authors were granted access to the HPC resources of IDRIS (Institut du développement et des ressources en informatique scientifique) under the allocation n^o A0102A01664 attributed by GENCI (Grand équipement national de calcul intensif). For this work we were also granted access to the HPC resources of Aix-Marseille Université financed by the project Equip@Meso (n^o ANR-10-EQPX- 29-01). The authors thankfully acknowledge financial support granted by MAEDI (Ministère des Affaires étrangères et du développement international), MEN (Ministère de l'Education Nationale), MESRI (Ministère de l'enseignement supérieur, de la recherche et de l'innovation), DAAD (Deutscher Akademischer Austauschdienst) within the French-German program Procope, project FIFIT (Flapping Insect Flight In Turbulence). JR, TE gratefully acknowledge financial support from DFG, grant n^o SFB1029. TE expresses his gratitude to Fritz-Olaf Lehmann for his postdoc fellowship at Rostock University.

References

[1] S. Abhyankar, J. Brown, E. M Constantinescu, D. Ghosh, B. F. Smith, and H. Zhang. Petsc/ts: A modern scalable ODE/DAE solver library. *arXiv:1806.01437*, 2018.

[2] P. Angot, C. Bruneau, and P. Fabrie. A penalization method to take into account obstacles in incompressible viscous flows. *Numer. Math.*, 81:497–520, 1999.

[3] E. Arquis and J.-P. Caltagirone. Sur les conditions hydrodynamiques au voisinage d'une interface milieu fluide milieu poreux: application à la convection naturelle. *C. R. Acad. Sci. Paris, Sér. II*, 299, 1984.

[4] R. Becker and R. Rannacher. *An Optimal Control Approach to a-posteriori Error Estimation*, pages 1–102. Cambridge University Press, Cambridge, 2001.

[5] M. Berger and P. Colella. Local adaptive mesh refinement for shock hydrodynamics. *J. Comput. Phys.*, 82:64–84, 1988.

[6] M. Berger and J. Oliger. Adaptive mesh refinement for hyperbolic partial differential equations. *J. Comput. Phys.*, 53:484–512, 1984.

[7] A. Brandt. Multilevel adaptive solutions to boundary value problems. *Math. Comp.*, 31:333–390, 1977.

[8] C. Burstedde, L. C. Wilcox, and O. Ghattas. `p4est`: Scalable algorithms for parallel adaptive mesh refinement on forests of octrees. *SIAM J. Sci. Comput.*, 33(3):1103–1133, 2011.

[9] A.J. Chorin. A numerical method for solving incompressible viscous flow problems. *J. Comput. Phys.*, 2:12–26, 1967.

[10] A. Cohen. *Wavelet Methods in Numerical Analysis*, volume VII of *Handbook of Numerical Analysis*. Elsevier, Amsterdam, 2000.

[11] A. Cohen, W. Dahmen, and R. DeVore. Adaptive wavelet methods for elliptic operator equations – convergence rates. *Math. Comp*, 70:27–75, 1998.

[12] A. Cohen, I. Daubechies, and J. C. Feauveau. Biorthogonal bases of compactly supported wavelets. *Comm. Pure Appl. Math.*, 45:485–560, 1992.

[13] A. Cohen, N. Dyn, S. M. Kaber, and M. Postel. Multiresolution schemes on triangles for scalar conservation laws. *J. Comp. Phys.*, 161:264–286, 2000.

[14] I. Daubechies. Orthonormal bases of compactly supported wavelets. *Comm. Pure Appl. Math.*, 41(7):909–996, 1988.

[15] R. Deiterding. *Parallel Adaptive Simulation of Multi-Dimensional Detonation Structures*. PhD thesis, Brandenburgische Technische Universität Cottbus, 2003.

[16] R. Deiterding. Construction and application of an amr algorithm for distributed memory computers. In Tomasz Plewa, Timur Linde, and V. Gregory Weirs, editors, *Adaptive Mesh Refinement – Theory and Applications*, pages 361–372, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[17] R. Deiterding, M. Domingues, S. Gomes, O. Roussel, and K. Schneider. Adaptive multiresolution or adaptive mesh refinement? A case study for 2D Euler equations. *ESAIM: Proceedings*, 29:28–42, 2009.

[18] R. Deiterding, M. Domingues, S. Gomes, O. Roussel, and K. Schneider. Comparison of adaptive multiresolution and adaptive mesh refinement applied to simulations of the compressible Euler equations. *SIAM J. Sci. Comput.*, 38(5):S173–S193, 2016.

[19] G. Deslauriers and S. Dubuc. Interpolation dyadique, 1987.

[20] G. Deslauriers and S. Dubuc. Symmetric iterative interpolation processes. *Constr. Approx.*, 5(1):49, 1989.

[21] R. A. DeVore. Nonlinear approximation. *Acta Numerica*, 7:51–150, 1998.

[22] E. Dilek, B. Erzincanli, and M. Sahin. The numerical investigation of lagrangian and eulerian coherent structures for the near wake structure of a hovering drosophila. *Theoretical and Computational Fluid Dynamics*, Apr 2019.

[23] M. Domingues, S. Gomes, O. Roussel, and K. Schneider. Adaptive multiresolution methods. *ESAIM: Proceedings*, 34:1–96, 2011.

[24] M. Domingues, S. M. Gomes, and L. M. A. Díaz. Adaptive wavelet representation and differentiation on block-structured grids. *Appl. Num. Math.*, 47:421–437, 2003.

[25] D L Donoho. Interpolating wavelet transforms. *Preprint, Department of Statistics, Stanford University*, 2(3):1–54, 1992.

[26] D. L. Donoho and J. M. Johnstone. Ideal spatial adaptation by wavelet shrinkage. *Biometrika*, 81(3):425–455, 1994.

[27] M. Dreissigacker. Turbulence generated by fractal trees – PIV measurements and comparison with numerical data. Master's thesis, TU Berlin, 2017.

[28] T. Engels, D. Kolomenskiy, K. Schneider, M. Farge, F.-O. Lehmann, and J. Sesterhenn. Helical vortices generated by flapping wings of bumblebees. *Fluid Dyn. Res.*, 50:011419, 2018.

[29] T. Engels, D. Kolomenskiy, K. Schneider, M. Farge, F.-O. Lehmann, and J. Sesterhenn. Impact of turbulence on flying insects in tethered and free flight: High-resolution numerical

experiments. *Phys. Rev. Fluids*, 4:013103, Jan 2019.

[30] T. Engels, D. Kolomenskiy, K. Schneider, F.-O. Lehmann, and J. Sesterhenn. Bumblebee flight in heavy turbulence. *Phys. Rev. Lett.*, 116:028103, 2016.

[31] T. Engels, D. Kolomenskiy, K. Schneider, and J. Sesterhenn. Two-dimensional simulation of the fluttering instability using a pseudospectral method with volume penalization. *Computers & Structures*, 122:101–112, 2012.

[32] T. Engels, D. Kolomenskiy, K. Schneider, and J. Sesterhenn. Numerical simulation of fluid-structure interaction with the volume penalization method. *J. Comput. Phys.*, 281:96–115, 2015.

[33] T. Engels, D. Kolomenskiy, K. Schneider, and J. Sesterhenn. FluSI: A novel parallel simulation tool for flapping insect flight using a Fourier method with volume penalization. *SIAM J. Sci. Comput.*, 38(5):S3–S24, 2016.

[34] M. Farge, K. Schneider, and N. Kevlahan. Non-Gaussianity and Coherent Vortex Simulation for two-dimensional turbulence using an adaptive orthonormal wavelet basis. *Phys. Fluids*, 11:2187–2201, 1999.

[35] I. Gargantini. An effective way to represent quadtrees. *Communications of the ACM*, 25(12):905–910, 1982.

[36] I. Gargantini. Linear octtrees for fast processing of three-dimensional objects. *Computer graphics and Image processing*, 20(4):365–374, 1982.

[37] A. Grossmann and J. Morlet. Decomposition of Hardy functions into square integrable wavelets of constant shape. *SIAM J. Math. Anal.*, 15:723–736, 1984.

[38] J. Guermond and P. Minev. High-order time stepping for the incompressible Navier–Stokes equations. *SIAM J. Sci. Comput.*, 37(6):A2656–A2681, 2015.

[39] A. Harten. Discrete multi-resolution analysis and generalized wavelets. *Appl. Numer. Anal.*, 12:153–192, 1993.

[40] A. Harten. Multiresolution algorithms for the numerical solution of hyperbolic conservation laws. *Comm. Pure Appl. Math.*, 48:1305–1342, 1995.

[41] A. Harten. Multiresolution representation of data: A general framework. *SIAM J. Numer. Anal.*, 33:385–394, 1996.

[42] D. Hilbert. Über die stetige Abbildung einer Linie auf ein Flächenstück. *Mathematische Annalen*, 38:459–460, 1891.

[43] M. Holmström. Solving hyperbolic PDEs using interpolating wavelets. *SIAM J. Sci. Comput.*, 21(2):405–420, 1999.

[44] N. Kevlahan and M. Farge. Vorticity filaments in two-dimensional turbulence: Creation, stability and effect. *J. Fluid Mech.*, 346:49–76, 1997.

[45] N. Kevlahan and O. Vasilyev. An adaptive wavelet collocation method for fluid-structure interaction at high Reynolds numbers. *SIAM J. Sci. Comput.*, 26(6):1894–1915, 2005.

[46] D. Kolomenskiy, J.-C. Nave, and K. Schneider. Adaptive gradient-augmented level set method with multiresolution error estimation. *J. Sci. Comput.*, 66:116–140, 2016.

[47] D. Kolomenskiy and K. Schneider. A Fourier spectral method for the Navier-Stokes equations with volume penalization for moving solid obstacles. *J. Comput. Phys.*, 228:5687–5709, 2009.

[48] J. Liandrat and P. Tchamitchian. Resolution of the 1d regularized Burgers equation using a spatial wavelet approximation. ICASE Technical report, 1990.

[49] S. Mallat. Multiresolution approximation and orthonormal wavelet basis of $L^2(\mathbb{R})$. *Trans. Am. Math. Soc.*, 315:69–87, 1989.

[50] S. Mallat. *A Wavelet Tour of Signal Processing the Sparse Way*. Academic Press, 3rd edition,

2009.

[51] R. Mittal and G. Iaccarino. Immersed boundary methods. *Annu. Rev. Fluid Mech.*, 37:239–261, 2005.

[52] S. Müller. *Adaptive Multiscale Schemes for Conservation Laws*, volume 27 of *Lectures Notes in Computational Science and Engineering*. Springer, Heidelberg, 2003.

[53] R. Nguyen van yen, D. Kolomenskiy, and K. Schneider. Approximation of the Laplace and Stokes operators with Dirichlet boundary conditions through volume penalization: A spectral viewpoint. *Numer. Math.*, 128:301–338, 2014.

[54] T. Ohwada and P. Asinari. Artificial compressibility method revisited: Asymptotic numerical method for incompressible Navier–Stokes equations. *J. Comp. Phys.*, 229:1698–1723, 2010.

[55] C. S. Peskin. Numerical analysis of blood flow in the heart. *J. Comput. Phys.*, 25:220–252, 1977.

[56] C. S. Peskin. The immersed boundary method. *Acta Numerica*, 11:479–517, 2002.

[57] D. Rossinelli, C. Conti, and P. Koumoutsakos. Mesh-particle interpolations on graphics processing units and multicore central processing units. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 369(1944):2164–2175, May 2011.

[58] O. Roussel and K. Schneider. Coherent Vortex Simulation of weakly compressible turbulent mixing layers using adaptive multiresolution methods. *J. Comput. Phys.*, 229:2267–2286, 2010.

[59] O. Roussel, K. Schneider, A. Tsigulin, and H. Bockhorn. A conservative fully adaptive multiresolution algorithm for parabolic PDEs. *J. Comput. Phys.*, 188:493–523, 2003.

[60] K. Schneider. Immersed boundary methods for numerical simulation of confined fluid and plasma turbulence in complex geometries: a review. *J. Plasma Phys.*, 81:435810601, 2015.

[61] K. Schneider and O. Vasilyev. Wavelet methods in computational fluid dynamics. *Annu. Rev. Fluid Mech.*, 42:473–503, 2010.

[62] F. Schornbaum and U. Rüde. Extreme-scale block-structured adaptive mesh refinement. *SIAM J. Sci. Comput.*, 40(3):C358–C387, 2018.

[63] 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.

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

[65] W. Sweldens and P. Schröder. Building your own wavelets at home. *Wavelets in Computer Graphics*, 1997.

[66] W. Sweldens and P. Schröder. *Building Your Own Wavelets at Home*, pages 72–107. Springer Berlin Heidelberg, Berlin, Heidelberg, 2000.

[67] 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.

[68] H. Truong, T. Engels, D. Kolomenskiy, and K. Schneider. A mass-spring fluid-structure interaction solver: Application to flexible revolving wings. *Comput. & Fluids*, 200:104426, 2020.

[69] G. Zumbusch. *Parallel Multilevel Methods: Adaptive Mesh Refinement and Loadbalancing.* Advances in Numerical Mathematics. Springer, 2003.

Electronic Supplementary Material for

A Wavelet-Adaptive Method for Multiscale Simulation of Turbulent Flows in Flying Insects

Thomas Engels[1,4,*], Kai Schneider[2], Julius Reiss[3] and Marie Farge[4]

[1] *Institute of Biosciences, University of Rostock, Rostock, Germany.*
[2] *Aix-Marseille Université, CNRS, I2M UMR 7373, Marseille, France.*
[3] *ISTA, Technische Universität Berlin, Berlin, Germany.*
[4] *LMD UMR 8539 École Normale Supérieure-PSL, Paris, France.*

A Approximation for 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

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

where $a_j^{(\mathrm{4th})}$ and $a_j^{(\mathrm{6th})}$ 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

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

B Design of the computational grid

The design of our adaptive grid is tailored to the requirements of biorthogonal wavelets (nestedness) and designed with the idea of keeping the numerical code as simple as possible. We therefore use blocks of equal size B_s^D, where D is the problems dimensionality, as explained in the main text. Still, the precise definition of the grid is not unique using these requirements, which is why we justify our choices in detail in this appendix.

---

[*] Corresponding author. *Email addresses:* `thomas.engels@ens.fr` (T. Engels), `kai.schneider@univ-amu.fr` (K. Schneider), `julius.reiss@tnt.tu-berlin.de` (J. Reiss), `marie.farge@ens.fr` (M. Farge)

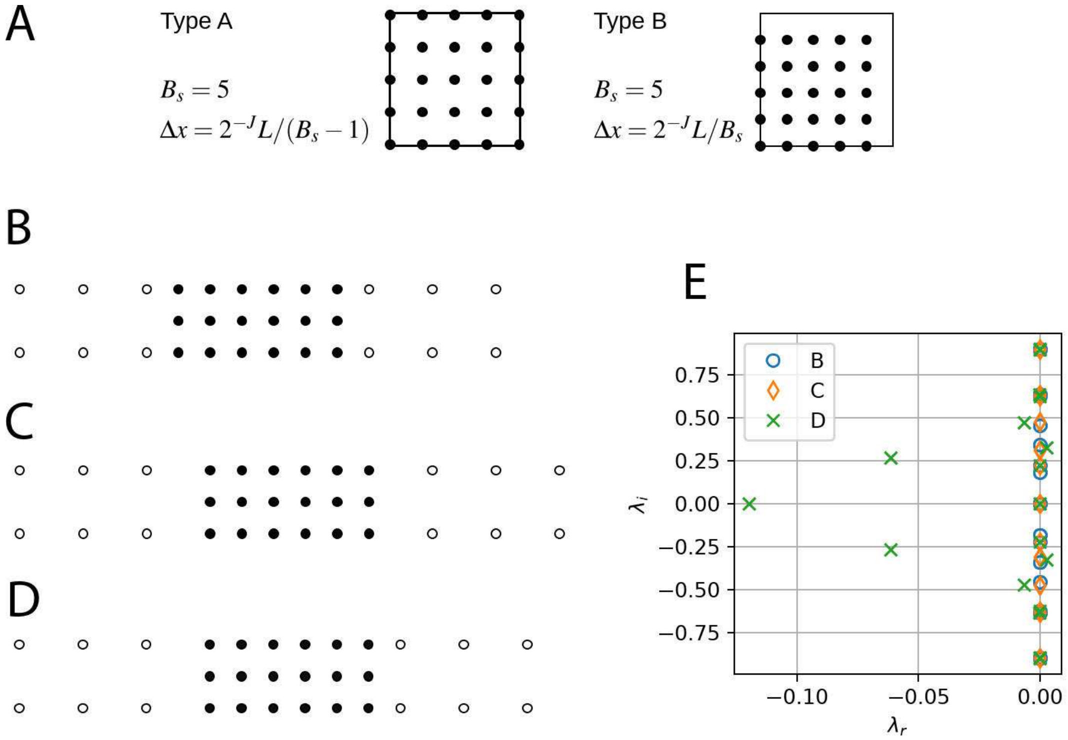


Figure 1: A: Two possible block definitions using the same block size B_S. Type A includes all of the block borders, which are then redundant with the neighboring block, while type B assigns each block border to one block. Present work uses type A grids exclusively. B-D: Different grid definitions for 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.

Fig. 1A illustrates the two possibilities to define a block, where type A blocks include all borders and type B only half of them. Consequently, using type A, two adjacent blocks sharing a common interface both contain redundant points, which is not the case using blocks of type B. At first it may thus seem that the latter are preferable.

Different discretizations of the block interface with a jump in resolution are shown in Fig. 1B-D. The spacing between the block is equal to the fine (B) and coarse (C) block spacing, and symmetrical on both sides. In D, both are mixed. The eigenvalues of the discrete finite difference operator for the first derivative are shown in E. Clearly, grids B,C result in purely imaginary eigenvalues, while grid D also has non-zero real parts. While these are negative in the example shown, it depends on the local flow direction if the solution is excited or attenuated. From these eigenvalues, we hence conclude that grid D is unstable without an additional stabilization (e.g., filtering). We immediately note that grid D corresponds to a block-based grid using type B blocks, which is why we do not use this definition.

Blocks of type A allow stable discretizations shown in B,C, provided we attribute the points on the border of a block consistently to either the fine or coarse block. This final decision was made based on the observation that using CDF 4/4 wavelets, spurious oscillations are generated at resolution jumps if the interface is attributed to the fine block. These artifacts disappear if the interface is counted to the coarse block (grid C in Fig. 1). However, strictly speaking this grid definition is incompatible with the requirement for blocks of equal size. On the finer block, we therefore simply overwrite the solution with the coarse-block data at their common interface. This is done when synchronizing the ghost node layer. When using CDF 4/0 wavelets, as is the case for the 3D simulations in the main text, we attribute the points on the block/block borders to the fine block.

C CDF wavelets

For completeness, the filter coefficients for the biorthogonal wavelets used in this work (CDF 4/0 and CDF 4/4) are given in Table 1. The resulting scaling functions ϕ, wavelets ψ and dual functions $\widetilde{\phi}$, $\widetilde{\psi}$, are illustrated in Fig. 2.

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					$-1/256$			

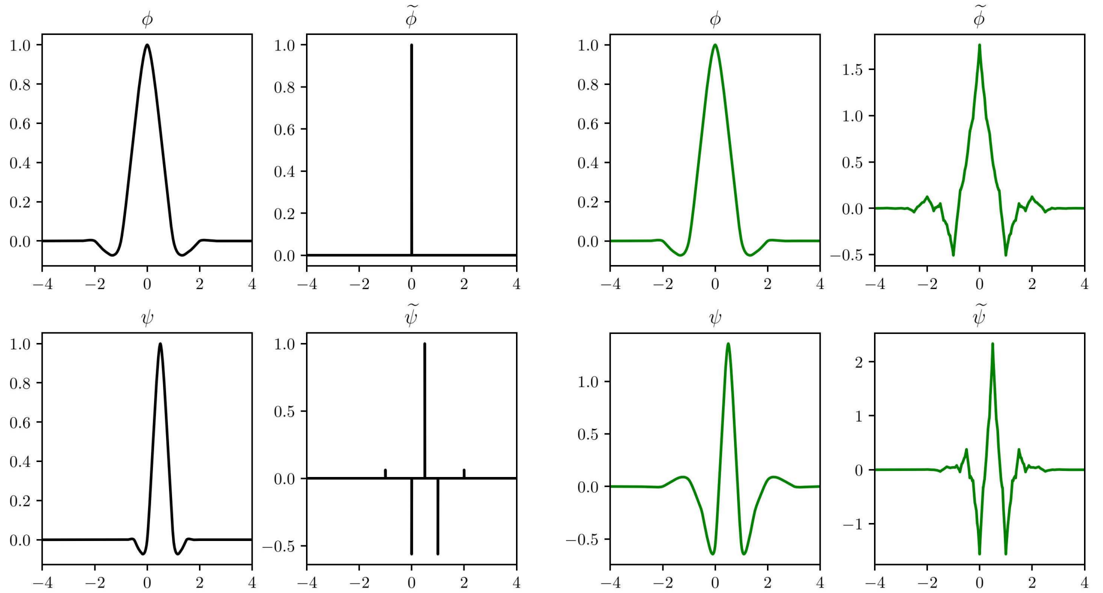


Figure 2: Scaling functions ϕ, wavelets ψ and dual functions $\widetilde{\phi}$, $\widetilde{\psi}$ for the CDF 4/0 (left, black) and CDF 4/4 wavelets (right, green).

D Non-reflecting boundary conditions using sponges

At the outflow boundary of the computational domain, we impose non-reflecting boundary conditions using a sponge technique. The outflow removes two physically distinct phenomena: on the one hand, vortical structures not yet dissipated, and artificial sound waves on the other. The artificial sound waves transport flow divergence created at the fluid–solid interface and due to the nonlinear term. It is possible to implement such conditions based on characteristics, but for simplicity and consistency with the treatment of the obstacles, we use the volume penalization method to include non-reflecting sponges. We can conveniently discuss the choice of parameters for sponges using a simple, one-dimensional model. In the acoustic regime, we can neglect the nonlinear and diffusion terms, and we also remove the obstacle penalization term, yielding

$$\partial_t u = -\partial_x p - \frac{\chi_{\mathrm{sp}}}{C_{\mathrm{sp}}} u,$$
$$\partial_t p = -\partial_x u - \frac{\chi_{\mathrm{sp}}}{C_{\mathrm{sp}}} p,$$

where we further normalize $C_0 = 1$ without loss of generality. The reference state is $u_\infty = p_\infty = 0$. We consider a periodic domain of length L and the initial condition

$$u(x, t=0) = \exp(-(x-L/2)^2/\sigma^2),$$
$$p(x, t=0) = 0.$$

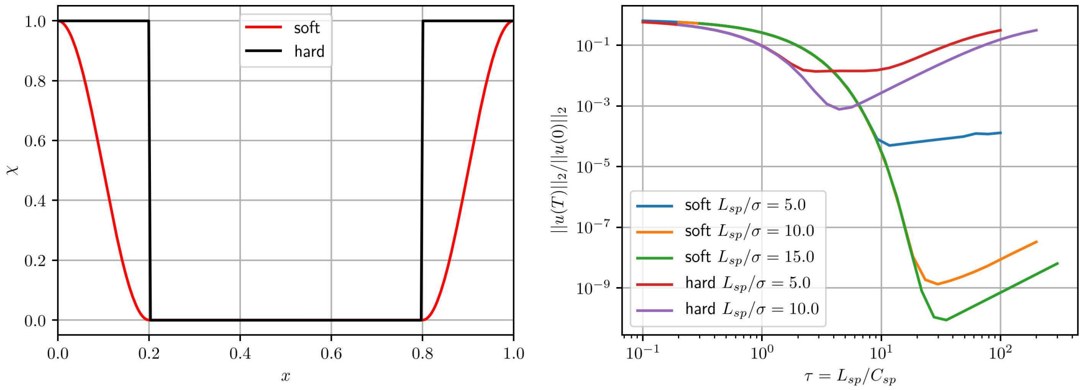


Figure 3: Left: 'Hard' and 'soft' χ_{sp} functions, here with $L_{sp}=0.2$. The red function is cos-shaped. Right: sponge error, computed as L^2 norm of residual velocity at $t=T$. Because the exact solution is zero, we use $\|u(t=0)\|_2$ as normalization.

This initial condition results in two artificial sound waves with characteristic size σ traveling either left or right. The waves do not decay in a 1D setting, unlike in higher dimensions, making the 1D problem the most sensitive to non-reflecting boundary conditions. In 3D, the waves decay as R^{-2}, where R is the distance to the source. Provided a sufficiently large domain, pseudoacoustic waves thus are already significantly weakened when reaching the sponge layer.

The size of the sponge layer should be larger than the waves it is supposed to absorb – the important ratio is thus L_{sp}/σ. The second relevant parameter is the time spent in the sponge layer, relative to the relaxation time C_{sp}. We call this parameter $\tau = L_{sp}/(C_0 C_{sp})$. The third relevant parameter is the shape χ_{sp} of the sponge layer, for which we can use either a discontinuous (hard) or smoothed (soft) function, as shown in Fig. 3(left).

Depending of the choice of parameters, one of two things can happen: the damping is too weak, and the wave is not completely removed from the domain (it re-enters on the opposite side). The damping can also be too strong, in which case the sponge term deteriorates the regularity of the solution and spurious reflections occur.

We run the simulations until $T=3L/4$; the artificial wave reaches the sponge around $T_0=(L-L_{sp})/2$. The exact solution at $t=T$ is $u=p=0$. Fig. 3(right) shows the normalized residual velocity at the final time T for hard and soft mask function and different L_{sp}/σ, as a function of τ. The curves coincide for small τ, which indicates that L_{sp}/σ and τ are indeed the relevant parameters. Thicker sponges allow setting larger values of τ, which reduces reflections. However, a significant improvement of the soft mask function over the discontinuous one is observed, and almost perfect cancellation of waves is observed for the largest L_{sp}/σ. From this data, we conclude that $L_{sp}/\sigma > 5$ and $\tau > 10$ minimize reflections. Sponges are usually set much larger than $L_{sp}/\sigma=5$, because of vortical structures. We recommend setting $\tau = 20$ in the main text. Note that in practice, simulations are performed in 3D and the sensitivity to the sponge is then significantly reduced.

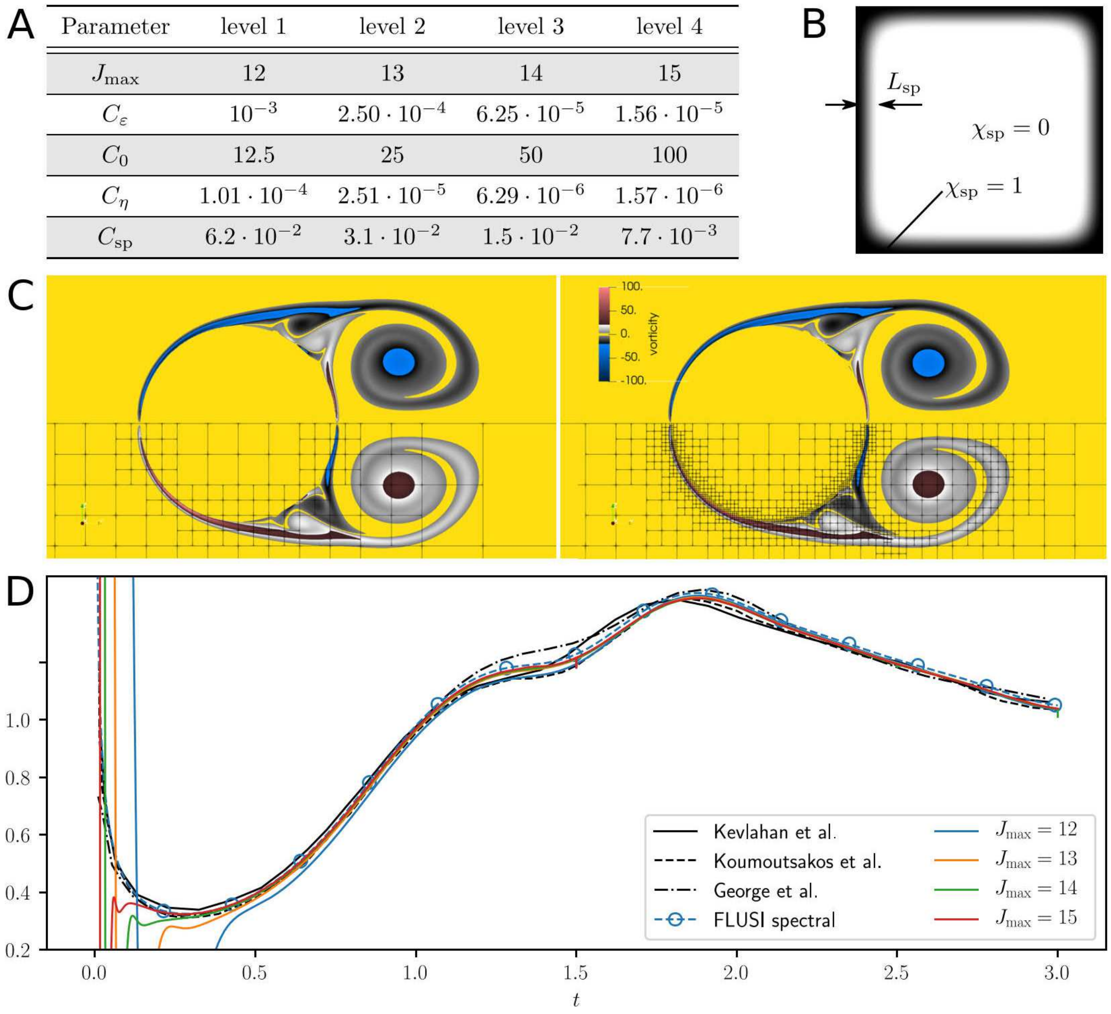
E Impulsively started cylinder

As benchmark we consider an impulsively started cylinder at $Re = Du_\infty/\nu = 3000$ and compare with the results in [2–4]. The volume penalization method is used to approximate the Dirichlet boundary condition, and the simulations are two-dimensional. A significant improvement with respect to spectral simulations is that the adaptive framework allows for setting very large domains (for the cylinder simulations, we use here $L = 160$), because the strongest refinement is located near the cylinder and a coarse grid is used in the surrounding domain. The flow is impulsively started, *i.e.*, the initial velocity is $\underline{u}(\underline{x}, t = 0) = \underline{u}_\infty$, where $\underline{u}_\infty = (1,0)^T$, and $p(\underline{x}, 0) = 0$. Four levels of resolution are performed, with the key parameters summarized in Fig. 4A. Because of the large computational domain, the coarsest simulation already uses $J_{\max} = 12$ scales.

Fig. 4C shows the vorticity field at the final time $t = 3.0$ for the lowest (12) and highest (15) maximum resolution level $J_{\max}$. The vorticity is visually very similar to results presented in the literature [2–4], and the two simulations show only little differences. Two main vortices are visible behind the cylinder, and secondary and tertiary vorticity lead to a dipole-like flow pattern near the separation point [4].

The evolution of the drag coefficient is shown in Fig. 4D together with the reference data and an incompressible spectral simulation with penalization [1] (using $L = 20$, 16384×16384 modes, $K_\eta = 0.1$). During the impulsive startup phase, present results obtained with ACM differ as expected significantly from the reference solutions. While the force is very large in the beginning, it later decreases and even becomes negative. After some time, depending on C_0, the drag however approaches the reference curves. In our applications to insect flight, impulsive start is used because of the lack of suitable initial conditions, rather than an interest in the short-time behavior of the solution. If one is interested in this transient behavior, the initial speed of sound C_0 should be very large and can be decreased after some time. Another possibility is to follow Ohwada [5] and carefully derive appropriate initial conditions. For our focus on flapping flight, this is not required.

As discussed, the impulsive start results in pressure waves propagating in the domain, and they can contribute significantly to the number of blocks used in the simulation. The pattern is similar in all simulations: the initial wave entails an increasing N_b, until the wave is absorbed by the sponge (Fig. 4B). Then, the grid density rapidly decreases and then remains practically constant. In all cases, less than 0.1% of the corresponding full grid is used at most, which is also related to the large domain size. Note that pressure waves decay much faster in 3D than in 2D, as explained in the main text.



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}	$6.2 \cdot 10^{-2}$	$3.1 \cdot 10^{-2}$	$1.5 \cdot 10^{-2}$	$7.7 \cdot 10^{-3}$

Figure 4: Impulsively started cylinder at Re=3000. A: computational parameters for the four levels of resolution. B: mask function χ_{sp} for the sponge. The actual cylinder is not visible due to the large domain size. C: vorticity field with the block-based grid superimposed in the lower half, results from the level 1 and 4 computations, respectively. D: drag coefficient as a function of time, compared with results from the literature. All results are obtained with CDF 4/0 wavelets.

F Description of parameter files required to reproduce simulations

This section summarizes and briefly explains the content of the electronic supplementary material (ESM). The intention of the ESM is to ensure reproducibility of all results presented in the main text.

General remarks on running the code

The `WABBIT` code is started from a jobfile specific to the supercomputer. The general call to the code is `MPI_COMMAND ./wabbit PARAMETERS.ini -memory=250.0gb`. The parameter file might refer to other parameter files (for example for the wingbeat kinematics or the fractal tree).

Three-vortices simulations

Files required to run a simulation:

- Main parameter file: `PARAMS-three-vortices.ini`
- Initial condition:
  `ux_000000000000.h5`, `uy_000000000000.h5`, `p_000000000000.h5`

As we present a total of 160 simulations for this problem, we decided to not include all required parameter files. The ESM rather contains a single simulation with CDF 4/4 wavelets, $J_{\max}=5$, $C_0=10$ and $C_\varepsilon=2.6827\cdot10^{-4}$.

Flapping wing simulations: Suzuki test case

Files required to run a simulation:

- Main parameter file: `PARAMS_suzuki_jmax5.ini`
  (or `*_jmax6.ini`, `*_jmax7.ini`)

Each of the coarse, medium and fine resolution simulation is included in the ESM.

Bumblebee simulations

Files required to run a simulation:

- Main parameter file: `PARAMS_bumblebee_jmax*.ini`
- Wingbeat kinematics file: `bumblebee_new_kinematics.ini`

Each of the coarse, medium and fine resolution simulation is included in the ESM.

Bumblebee and fractal tree

Files required to run the simulation:

- Main parameter file: `PARAMS_bumblebee_fractaltree.ini`
- Wingbeat kinematics file: `bumblebee_new_kinematics.ini`
- Text file describing the cylinders in the fractal tree: `tree_SPH_branch2.in`

References

[1] T. Engels, D. Kolomenskiy, K. Schneider, and J. Sesterhenn. FluSI: A novel parallel simulation tool for flapping insect flight using a Fourier method with volume penalization. *SIAM J. Sci. Comput.*, 38(5):S3–S24, 2016.

[2] A. George, L. C. Huang, W.-P. Tang, and Y. D. Wu. Numerical simulation of unsteady incompressible flow ($Re \leq 9500$) on the curvilinear half-staggered mesh. *SIAM J. Sci. Comput.*, 21(6):2331–2351, 2000.

[3] N. Kevlahan and O. Vasilyev. An adaptive wavelet collocation method for fluid-structure interaction at high Reynolds numbers. *SIAM J. Sci. Comput.*, 26(6):1894–1915, 2005.

[4] P. Koumoutsakos and A. Leonard. High-resolution simulations of the flow around an impulsively started cylinder using vortex methods. *J. Fluid Mech.*, 296:1–38, 1995.

[5] T. Ohwada and P. Asinari. Artificial compressibility method revisited: Asymptotic numerical method for incompressible Navier–Stokes equations. *J. Comp. Phys.*, 229:1698–1723, 2010.

[6] 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.