SIAM J. SCI. COMPUT.
Vol. 38, No. 5, pp. S3–S24

FluSI: A NOVEL PARALLEL SIMULATION TOOL FOR FLAPPING INSECT FLIGHT USING A FOURIER METHOD WITH VOLUME PENALIZATION*

THOMAS ENGELS†, DMITRY KOLOMENSKIY‡, KAI SCHNEIDER§, AND JÖRN SESTERHENN¶

Abstract. We introduce FluSI, a fully parallel open source software package for pseudospectral simulations of three-dimensional flapping flight in viscous flows. It is freely available for noncommercial use from GitHub (https://github.com/pseudospectators/FLUSI). The computational framework runs on high performance computers with distributed memory architectures. The discretization of the three-dimensional incompressible Navier–Stokes equations is based on a Fourier pseudospectral method with adaptive time stepping. The complex time varying geometry of insects with rigid flapping wings is handled using the volume penalization method. The modules characterizing the insect geometry, flight mechanics, and wing kinematics are described. Validation tests for different benchmarks illustrate the efficiency and precision of the approach. Finally, computations for a model insect in the turbulent regime demonstrate the versatility of the software.

Key words. pseudospectral method, volume penalization, flapping insect flight

AMS subject classifications. 76M22, 65M85, 74F10, 76Z10, 65Y05, 76F65

DOI. 10.1137/15M1026006

1. Introduction.

Flapping flight is an active interdisciplinary research field with many open questions, and numerical simulations have become an important instrument for tackling them. Here, we present the computational solution environment `FluSI`, which is, to the best of our knowledge, the first open source code in this field.

In the literature different numerical strategies have been proposed for simulating flapping flight, e.g., the use of several overlapping grids [19], which are adapted to the geometry, or the use of moving grids with the arbitrary Lagrangian–Eulerian method [29]. Those methods involve significant computational and implementation overhead, the reduction of which has motivated the development of methods that allow for geometry-independent discretization, such as immersed boundary methods.

In this work, we employ the volume penalization method to take the no-slip boundary condition into account. The idea of modeling solid obstacles as porous media with a small permeability was proposed in [2]. The forcing term acts on the entire volume of the solid and not only on its surface, as it is the case in the immersed boundary methods, and corresponds to the Darcy drag. The distinctive feature of the

*Received by the editors June 15, 2015; accepted for publication (in revised form) May 6, 2016; published electronically October 27, 2016. The work of the first, third, and fourth authors was partially supported by the French-German University.

http://www.siam.org/journals/sisc/38-5/M102600.html

†Corresponding author. ISTA, Technische Universität Berlin, Müller-Breslau-Strasse 12, 10623 Berlin, Germany, and ENS Paris, Laboratoire de Météorologie Dynamique, 24, Rue Lhomond, 75231 Paris Cedex 05, France (thomas.engels@mailbox.tu-berlin.de).

‡Graduate School of Engineering, Chiba University, 1-33, Yayoi-cho, Inage-ku, Chiba-shi, Chiba, 263-8522, Japan (dkolom@gmail.com). This author's work was partially supported by the Japan Society for the Promotion of Science through a postdoctoral fellowship (JSPS KAKENHI grant 15F15061).

§I2M-CNRS, Centre de Mathematiques and d'Informatique, Aix-Marseille Université, 39 rue Joliot-Curie, 13453 Marseille cedex 13, France (kschneid@cmi.univ-mrs.fr).

¶ISTA, Technische Universität Berlin, Müller-Breslau-Strasse 12, 10623 Berlin, Germany (joern.sesterhenn@tu-berlin.de).

method is the existence of rigorous convergence proofs [1, 6] showing that the solution of the penalized equations does indeed converge to the solution of the Navier–Stokes equations with no-slip boundary conditions. This approach has been extended to model not only Dirichlet conditions applied at the surface of moving, rigid, and flexible obstacles [17, 9], but also to homogeneous Neumann conditions, which is relevant for studying, e.g., turbulent mixing of a passive scalar [15]. This technique can potentially be useful for numerical simulation of odor-modulated navigation, as the odor can be treated as an essentially passive scalar. An interesting recent development was proposed in [13] in the context of finite-difference discretizations. Their idea is to modify the fractional step projection scheme such that the Neumann boundary condition, as introduced in [15], appears in the pressure Poisson equation. Another variant, specifically adapted to impulsively started flow, is the iterative penalization method proposed in [12]. For reviews on immersed boundary and penalization techniques, we refer the reader to [22, 27, 32].

The remainder of this article is organized as follows. First we discuss, in section 2, the numerical method of `FluSI`'s fluid module, which is based on a spectral discretization of the three-dimensional penalized Navier–Stokes equations. Section 3 describes the module that creates the insect geometry, as well as the governing equations for free flight simulations, and we focus on the parallel implementation in section 4. Thereafter we present a validation of our software solution environment in section 5 for different test cases established in the literature, and give an outlook for applications to flapping flight in the turbulent regime in section 6. Conclusions are drawn in section 7.

2. Numerical method.

2.1. Model equations. For the numerical simulation of many flapping fliers, like insects, the fluid through which they move can typically be approximated as incompressible. It is thus governed by the incompressible Navier–Stokes equations with the no-slip boundary condition on the fluid-solid interface. The numerical solution of this problem poses two major challenges. First, the pressure is a Lagrangian multiplier that ensures the divergence-free condition. Traditional numerical schemes employ a fractional step or projection method [16], requiring the solution of a Poisson problem for the pressure. Second, the no-slip boundary condition must be satisfied on a possibly complicated and moving fluid-solid interface, for which boundary-fitted grids are classical approaches [35]. Since the generation of these grids may be challenging, alternatives have been developed. The principal idea is to extend the computational domain to the interior of obstacles. The boundary is then taken into account by adding supplementary terms to the Navier–Stokes equation. The technique chosen here is the volume penalization method, which is physically motivated by replacing the solid obstacle by a porous medium with small permeability C_η. The penalized version of the Navier–Stokes equations then reads

$$\partial_t \underline{u} + \underline{\omega} \times \underline{u} = -\nabla \Pi + \nu \nabla^2 \underline{u} - \underbrace{\frac{\chi}{C_\eta}(\underline{u} - \underline{u}_s)}_{\text{penalization}} - \underbrace{\frac{1}{C_{\text{sp}}} \nabla \times \frac{(\chi_{\text{sp}} \underline{\omega})}{\nabla^2}}_{\text{sponge}}, \tag{2.1}$$

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

$$\underline{u}(\underline{x}, t = 0) = \underline{u}_0(\underline{x}), \qquad \underline{x} \in \Omega, t > 0, \tag{2.3}$$

where $\underline{u}$ is the fluid velocity, $\underline{\omega} = \nabla \times \underline{u}$ is the vorticity, and ν is the kinematic viscosity. The sponge term is explained in section 2.3. Equations (2.1)–(2.3) are written in dimensionless form, and the fluid density ϱ_f is normalized to unity. The nonlinear term in (2.1) is written in the rotational form; hence one is left with the gradient of the total pressure $\Pi = p + \frac{1}{2}\underline{u} \cdot \underline{u}$ instead of the static pressure p [28]. This formulation is chosen because of its favorable properties when discretized with spectral methods, namely conservation of momentum and energy [28, p. 210]. At the exterior of the computational domain, which is supposed to be sufficiently large, periodic boundary conditions can be assumed. The mask function χ is defined as

$$\chi(\underline{x}, t) = \begin{cases} 0 & \text{if } \underline{x} \in \Omega_f, \\ 1 & \text{if } \underline{x} \in \Omega_s, \end{cases} \tag{2.4}$$

where Ω_f is the fluid and Ω_s the solid domain. In anticipation of the application to moving boundaries, we note that we will replace the discontinuous χ-function by a smoothed one, with a thin smoothing layer centered around the interface. The mask function encodes all geometric information about the problem, and (2.1)–(2.3) do not include no-slip boundary conditions.

Note that in the fluid domain Ω_f one recovers the original equation as the penalization term $\frac{\chi}{C_\eta}(\underline{u} - \underline{u}_s)$ vanishes. The convergence proof in [6, 1] shows that the solution of the penalized Navier–Stokes equations (2.1)–(2.3) indeed tends for $C_\eta \to 0$ towards the exact solution of Navier–Stokes equation, imposing no-slip boundary conditions, with a convergence rate of $\mathcal{O}\left(\sqrt{C_\eta}\right)$ in the L^2-norm. The parameter C_η should thus be chosen small enough, which is also intuitively clear by the physical interpretation of C_η as permeability. However, the choice of C_η is subject to constraints, as the penalized equations are discretized and solved numerically. The modeling error of order $\mathcal{O}\left(\sqrt{C_\eta}\right)$ should be of the same order as the discretization error [24]. It is first noted that in (2.1), C_η has the dimension of time. It is instructive to put the nonlinear, viscous, and pressure terms aside for a moment. One is then left with $\partial_t \underline{u} = -\underline{u}/C_\eta$ inside the solid, with the obvious solution $\underline{u} = \underline{u}_0 \exp(-t/C_\eta)$. Thus, C_η can be directly identified as the relaxation time. Interfering with the time step Δt, usually implying $\Delta t = \mathcal{O}(C_\eta)$, this simple fact has important consequences for the numerical solution. It indicates that a good choice for C_η is not only "small enough," but also "as large as possible." Further insight into the properties of the penalization method can be obtained considering the penalization boundary layer of width $\delta_\eta = \sqrt{\nu C_\eta}$, which forms inside the obstacle Ω_s, as shown in [6]. When increasing the resolution, the number of points per boundary layer thickness, $K = \delta_\eta/\Delta x$, should be kept constant, which implies the relation $C_\eta \propto (\Delta x)^2$, consistent with [24], where the penalized Laplace and Stokes operators were analyzed analytically and numerically. With the scaling for C_η, one still has to choose the constant K. In fact, for any value of K, the method will converge with the same convergence rate, but the error offset can be tuned. For the two-dimensional Couette flow, in [9] we reported the optimal value of $K = 0.128$. In a range of numerical validation tests, including the ones presented here, we find $K = 0.1$–0.4 as a good choice which we recommend as a guideline for practical applications.

To satisfy the incompressibility constraint (2.2), a Poisson equation for the pressure is derived by taking the divergence of (2.1), yielding

$$\nabla^2 \Pi = \nabla \cdot \left(-\underline{\omega} \times \underline{u} - \frac{\chi}{C_\eta}(\underline{u} - \underline{u}_s) \right). \tag{2.5}$$

By construction of the method, (2.5) can be solved with periodic boundary conditions.

The aerodynamic force $\underline{F}$ and the torque moment $\underline{m}$, both important quantities in the present applications, can be computed from

$$\underline{F} = \oint_{\partial\Omega_s} \sigma \cdot \underline{n}\,\mathrm{d}\gamma = \frac{1}{C_\eta}\int_\Omega \chi\,(\underline{u}-\underline{u}_s)\,\mathrm{d}V + \frac{\mathrm{d}}{\mathrm{d}t}\int_{\Omega_s}\underline{u}_s \mathrm{d}V, \tag{2.6}$$

$$\underline{m} = \oint_{\partial\Omega_s} \underline{r}\times(\sigma\cdot\underline{n})\;\mathrm{d}\gamma = \frac{1}{C_\eta}\int_\Omega \underline{r}\times\chi\,(\underline{u}-\underline{u}_s)\,\mathrm{d}V + \frac{\mathrm{d}}{\mathrm{d}t}\int_{\Omega_s}\underline{r}\times\underline{u}_s \mathrm{d}V, \tag{2.7}$$

where the last terms in the right-hand sides are denoted "unsteady corrections" [36].

2.2. Penalization method for moving boundary problems. The volume penalization method as discussed so far assumes a discontinuous mask function in the form of (2.4). When applying the method to a nongrid aligned body, for example a circular cylinder, the mask function geometrically approximates the boundary to first order in Δx. Results for a stationary cylinder using the discontinuous mask function are acceptable [31, 33], but spurious oscillations in the case of moving bodies are reported [17]. The reason for this is that the discontinuous mask can be translated only by integer multiples of the grid spacing, and this jerky motion causes large oscillations in the aerodynamic forces. Kolomenskiy and Schneider [17] proposed an algorithm to shift the mask function in Fourier space instead of physical space. In the present work, we employ a different approach, because displacing the mask in Fourier space involves (additional) Fourier transforms, which are computationally expensive, especially if more than one rigid body is considered, as it is the case here. The idea on hand is to directly assume the mask function to be smoothed over a thin smoothing layer [8, 9]. To this end, we introduce the signed distance function $\delta(\underline{x},t)$ [25], and the mask function can then be computed from the signed distance, using

$$\chi(\delta) = \begin{cases} 1, & \delta \le -h, \\ \frac{1}{2}\left(1+\cos\left(\pi\frac{\delta+h}{2h}\right)\right), & -h<\delta<+h, \\ 0, & \delta > +h, \end{cases} \tag{2.8}$$

where the semithickness of the smoothing layer, h, is used. This is typically defined relative to the grid size, $h = C_{\mathrm{smth}}\Delta x$; thus (2.8) converges to a Heaviside step function as $\Delta x \to 0$. Nonetheless, it can be translated by less than one grid point, and then be resampled on the Eulerian fluid grid.

2.3. Wake removal techniques. The penalized Navier–Stokes equations (2.1)–(2.3) do not, by principle, include no-slip boundary conditions, and furthermore we discretize them in a periodic domain Ω. In such a periodic setting, the wake re-enters the domain, which is an undesired artifact. To overcome this issue, a supplementary "sponge" penalization term can be added to the vorticity-velocity formulation of the Navier–Stokes equations, in order to gradually damp the vorticity [8, 9]. The sponge penalization parameter C_{sp} is usually set to a larger value than C_η, typically $C_{\mathrm{sp}} = 10^{-1}$. The larger value and its longer relaxation time ensure that if a traveling vortex pair enters the sponge region, the leading one is not dissipated too fast, because it otherwise could leave the partner orphaned in the domain. Applying the Biot–Savart operator to the vorticity-velocity formulation, we formally find $-\frac{1}{C_{\mathrm{sp}}}\nabla\times\frac{(\chi_{\mathrm{sp}}\omega)}{\nabla^2}$. By construction the sponge term is divergence-free, which is important since otherwise it would contribute to the pressure, which in turn would be modified even in regions far away from the sponge due to its nonlocality. Moreover, this term leaves the mean

flow, i.e., the zeroth Fourier mode, unchanged. This technique is well adapted to spectral discretizations, since computing the sponge term requires the solution of three Poisson problems, which becomes a simple division in Fourier space. A discussion on the influence of the vorticity sponge can be found in section 6 (see Figure 6.3).

Dirichlet conditions on the velocity can be imposed directly with the volume penalization, which applies, for example, to channel walls. For simulations in a uniform, unbounded free-stream, we use both techniques; in a small layer at the domain borders, the Dirichlet condition $\underline{u} = \underline{u}_\infty$ is imposed with the same precision as the actual obstacle, and a preceding, thicker sponge layer ensures that the upstream influence is minimized [9]. The sponge technique is similar to the "fringe regions" proposed in [30]. However, in [30] only a velocity sponge is used, which corresponds to the penalization term, i.e., $-\chi_{\mathrm{sp}} \left(\underline{u} - \underline{u}_{\mathrm{sp}}\right) / C_{\mathrm{sp}}$. The vorticity sponge idea has the advantage of not requiring a "desired" velocity field $\underline{u}_{\mathrm{sp}}$, which has to be set a priori. It also does not contribute to the pressure, which helps reduce the region of influence.

2.4. Discretization. The model equations (2.1) and (2.5) can be discretized with any numerical scheme; in particular, a Fourier pseudospectral discretization can be used [31, 17, 9]. The general idea is to represent field variables as truncated Fourier series, and thus in three dimensions we have for any quantity φ (velocity, pressure, vorticity) the discrete complex Fourier coefficients $\widehat{\varphi}$. They can be computed efficiently with the fast Fourier transform (FFT) [7]. The gradient of a scalar q can be obtained by multiplying the Fourier coefficients $\widehat{q}$ with the wavevector $\underline{k} = (k_x, k_y, k_z)^T$ and the complex unit $\widehat{\nabla q} = \iota \underline{k} \widehat{q}$. The Laplace operator becomes a simple multiplication by $-|\underline{k}|^2$. When using, e.g., finite differences, the dominant part of the computational effort is spent on solving the Poisson equation in every time step [14]. This is a strong motivation for employing a Fourier discretization, as inverting a diagonal operator becomes a simple division. Inserting the truncated Fourier series into the model equations and requiring that the residual vanish with respect to all test functions (which are identical with the trial functions $\exp(\iota \underline{k} \cdot \underline{x})$) yields a Galerkin projection and results in an evolution equation for the Fourier coefficients of the velocity. The nonlinear and penalization terms contain products, which become convolutions in Fourier space. To speed up computation, the products are calculated in physical space. This last introduces aliasing errors which are virtually eliminated by the 2/3 rule [5], meaning that only 2/3 of the Fourier coefficients are retained. Such a mixture of spectral and physical computations is generally labeled "pseudospectral" and is, when dealiased, equivalent to a Fourier–Galerkin scheme. The code can be run with and without dealiasing, but our choice is to conservatively always turn dealiasing on. More details on the effect of dealiasing are discussed in [24, p. 318].

The spatially discretized equations can then be advanced in time in Fourier space. The code provides a classical fourth order Runge–Kutta scheme with explicit treatment of the diffusion term, and a second order Adams–Bashforth scheme with integrating factor [31].

Besides the fast solution of Poisson problems, the spectral method has the advantage of not adding numerical diffusion or dispersion to the penalized equation, in contrast to the case when discretized with finite differences. Furthermore, most of the computational effort is concentrated in the Fourier transforms, which is advantageous from a computational point of view. However, the discretization requires the use of an equidistant grid, which implies using a large number of grid points to resolve the thin boundary layers.

Note that we do not explicitly apply a Neumann-type boundary condition for

the pressure in our method. We use only periodic boundary conditions in the computational domain, which are natural for Fourier methods. It was shown in [1] that the pressure field of the penalized problem converges to the pressure in the original boundary value problem. The discretization that we use is consistent and stable, and therefore the numerical solution converges to the exact solution of the continuous penalized problem.

3. Virtual insect model. We previously described the fluid module where the geometry is taken into account by the penalization method, which is the interface with the insect module described next.

Insects fly by flapping their wings, which are basically flat with sharp edges and operate typically at a high angle of attack. In the following, we describe in detail the insect framework used in this work, the essential task being to construct χ and $\underline{u}_s$, which enter (2.1). The virtual insect consists of a body and two wings, all of which are performing solid body rotations around three axes and are assumed to be rigid. Therefore, we will make use of the rotation matrices

$$R_x(\xi) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\xi & \sin\xi \\ 0 & -\sin\xi & \cos\xi \end{pmatrix}, \qquad R_y(\xi) = \begin{pmatrix} \cos\xi & 0 & -\sin\xi \\ 0 & 1 & 0 \\ \sin\xi & 0 & \cos\xi \end{pmatrix},$$
$$R_z(\xi) = \begin{pmatrix} \cos\xi & \sin\xi & 0 \\ -\sin\xi & \cos\xi & 0 \\ 0 & 0 & 1 \end{pmatrix},$$

and define the different reference frames, namely the global $\underline{x}^{(g)}$, body $\underline{x}^{(b)}$, stroke plane $\underline{x}^{(s)}$, and wing $\underline{x}^{(w)}$, in which the geometry is defined. As described above, the mask function is constructed in each evaluation of the right-hand side as a function of the signed distance function, $\chi(\underline{x}) = \chi(\delta(\underline{x}))$, according to (2.8).

3.1. Body system. The insect's body is responsible for a major part of the total drag force. It is described by its logical center $\underline{x}^{(g)}_{\mathrm{cntr}}$, the translational velocity $\underline{u}^{(g)}_{\mathrm{cntr}}$, and the body angles β (pitch), γ (yaw), and ψ (roll); see Figure 3.1 The center point $\underline{x}^{(g)}_{\mathrm{cntr}}$ does not necessarily coincide with the center of gravity, but is rather an arbitrary point of reference. A point $\underline{x}^{(g)}$ in the global coordinate system can be transformed to the body system using the following linear transformation:

$$\underline{x}^{(b)} = M_{\mathrm{body}}(\psi,\beta,\gamma)\left(\underline{x}^{(g)} - \underline{x}^{(g)}_{\mathrm{cntr}}\right),$$
$$M_{\mathrm{body}} = R_x(\psi) R_y(\beta) R_z(\gamma). \tag{3.1}$$

Since rotation matrices do not commute, it is important to note that the body is first yawed, then pitched, and finally rolled, which is conventional in flight mechanics. The geometry of the body is defined in the body reference frame. The angular velocity of the body in the global system is

$$\underline{\Omega}^{(g)}_b = R_z^{-1}(\gamma)\left[\begin{pmatrix} 0 \\ 0 \\ \dot\gamma \end{pmatrix} + R_y^{-1}(\beta)\left[\begin{pmatrix} 0 \\ \dot\beta \\ 0 \end{pmatrix} + R_x^{-1}(\psi)\begin{pmatrix} \dot\psi \\ 0 \\ 0 \end{pmatrix}\right]\right],$$
$$\underline{\Omega}^{(b)}_b = M_{\mathrm{body}}\,\Omega^{(g)}_b,$$

which defines the velocity field inside the insect resulting from the body motion,

$$\underline{u}^{(g)}_b = \underline{u}^{(g)}_{\mathrm{cntr}} + M^{-1}_{\mathrm{body}}\left(\underline{\Omega}^{(b)}_b \times \underline{x}^{(b)}\right). \tag{3.2}$$

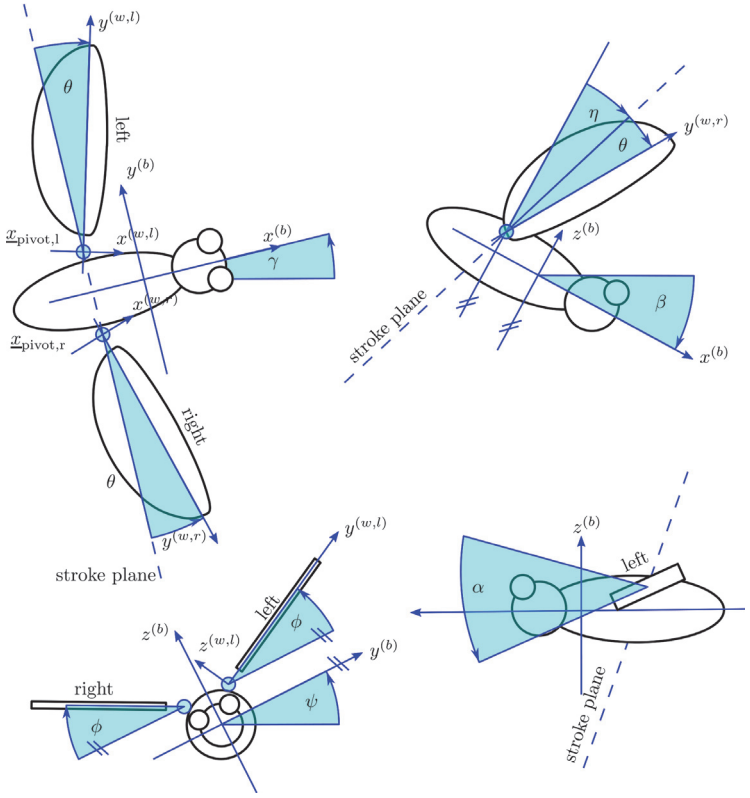

FIG. 3.1. *Model insect with definitions of the body angles γ (yaw), β (pitch), ψ (roll), the anatomical stroke plane angle η, the wing coordinate systems, and the wing angles θ (deviation), ϕ (position), and α (feathering). All angles are shown with positive sign.*

Equation (3.2) is valid also in the wings, since the flapping motion is prescribed relative to the body.

3.2. Body shape. The body shape is described in the body reference frame described previously. For instance, for the body depicted in Figure 3.1, which is composed of an ellipsoidal-shaped thorax and spheres for the head and eyes, the signed distance function is the intersection of the distance functions for the thorax, head, and eyes,

$$\delta_{\text{body}} = \max\left(\delta_{\text{thorax}}, \delta_{\text{head}}, \delta_{\text{eyes}}\right). \tag{3.3}$$

The max operator of the signed distances in (3.3) represents the intersection operator [25]. The signed distances for the components read

$$\begin{aligned}
\delta_{\text{thorax}}\left(\underline{x}^{(b)}\right) &= \sqrt{\left(y^{(b)}\right)^2 + \left(z^{(b)}\right)^2} - \sqrt{b^2\left(1 - \left(x^{(b)}/a\right)^2\right)}, \\
\delta_{\text{head}}\left(\underline{x}^{(b)}\right) &= \left|\underline{x}^{(b)} - \underline{x}^{(b)}_{0,\text{head}}\right|, \\
\delta_{\text{eyes}}\left(\underline{x}^{(b)}\right) &= \left|\underline{x}^{(b)} - \underline{x}^{(b)}_{0,\text{eyes}}\right|,
\end{aligned}$$

where a and b define the axes of the thorax ellipsoid and $\underline{x}^{(b)}_{0,\mathrm{head,eyes}}$ are the centers of the spheres.

3.3. Wing system. We consider only insects with two wings, one on each side, that are rotating about the pivot points $\underline{x}^{(b)}_{\mathrm{pivot,r}}$ and $\underline{x}^{(b)}_{\mathrm{pivot,l}}$. These pivots do not necessarily lie on the body surface; we rather allow a gap between wings and body. This gap avoids problems with nonsolenoidal velocity fields at the wing base. It is conventional to introduce a stroke plane, which is a plane tilted with respect to the body by an angle η. The coordinate in the stroke plane reads

$$\underline{x}^{(s)} = M_{\mathrm{stroke}} \left(\underline{x}^{(b)} - \underline{x}^{(b)}_{\mathrm{pivot}} \right).$$

Here, we use an anatomical stroke angle; that is, the angle η is defined relative to the body. Within the stroke plane, the wing motion is described by the angles α (feathering angle or angle of attack), ϕ (positional or flapping angle), and θ (deviation or out-of-stroke angle). Applying two rotation matrices yields the transformation from the body to the wing coordinate system:

$$\underline{x}^{(w)} = M_{\mathrm{wing}} \underline{x}^{(s)} = M_{\mathrm{wing}} M_{\mathrm{stroke}} \left(\underline{x}^{(b)} - \underline{x}^{(b)}_{\mathrm{pivot}} \right).$$

When flapping symmetrically, i.e., both wings following the same motion protocol, the stroke and wing rotation matrices for the left and right wing are given by

$$\begin{aligned} M_{\mathrm{stroke,l}} &= R_y(\eta), & M_{\mathrm{stroke,r}} &= R_x(\pi) R_y(\eta), \\ M_{\mathrm{wing,l}} &= R_y(\alpha) R_z(-\theta) R_x(\phi), & M_{\mathrm{wing,r}} &= R_y(-\alpha) R_z(-\theta) R_x(-\phi), \end{aligned}$$

where, due to the rotation $R_x(\pi)$, the sign of θ for the right wing does not have to be inverted. The angular velocities of the wings are given by

$$\begin{aligned} \underline{\Omega}^{(b)}_w &= M^{-1}_{\mathrm{stroke}} \left[R^{-1}_x(\phi) \left[\begin{pmatrix} \dot{\phi} \\ 0 \\ 0 \end{pmatrix} + R^{-1}_z(-\theta) \left[\begin{pmatrix} 0 \\ 0 \\ -\dot{\theta} \end{pmatrix} + R^{-1}_y(\alpha) \begin{pmatrix} 0 \\ \dot{\alpha} \\ 0 \end{pmatrix} \right] \right] \right], \\ \underline{\Omega}^{(w)}_w &= M_{\mathrm{wing}}\, \underline{\Omega}^{(b)}_w, \end{aligned}$$

which is used to compute the velocity field resulting from the wing motion,

$$\begin{aligned} \underline{u}^{(w)}_w &= \underline{\Omega}^{(w)}_w \times \underline{x}^{(w)}, \\ \underline{u}^{(g)}_w &= M^{-1}_{\mathrm{body}}\, M^{-1}_{\mathrm{wing}}\, M^{-1}_{\mathrm{stroke}}\, \underline{u}^{(w)}_w. \end{aligned} \tag{3.4}$$

The total velocity field inside the wings is given as the superposition of the body and wing rotation,

$$\underline{u}^{(g)}_s \left(\underline{x} \in \{\underline{x}_w\} \right) = \underline{u}^{(g)}_w + \underline{u}^{(g)}_b.$$

The actual kinematics, i.e., the angles $\alpha(t)$, $\phi(t)$, and $\theta(t)$, are parametrized by either Fourier or Hermite interpolation coefficients and read from a `*.ini` file.

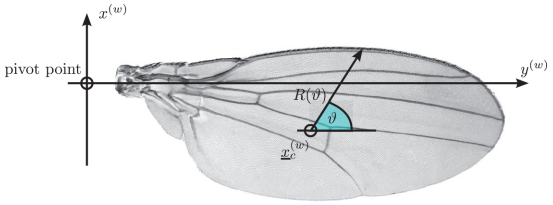

FIG. 3.2. *Realistic wing shapes are described in polar coordinates* $R(\vartheta)$.

3.4. Wing shape. In the previous section we defined the wing reference frame $\underline{x}^{(w)}$, in which we now describe the wing's signed distance function δ. In general, we define a set of several signed distance functions, each of which describes one surface of the wing. The signed distance function of the entire wing is then given by their intersection. For some model insects, we consider simple wings for which straightforward analytical expressions are available. For a rectangular wing, for instance the one illustrated later in Figure 5.2(c), we find for the signed distance function

$$\delta\left(\underline{x}^{(w)}\right) = \max\left(x^{(w)} - b; x^{(w)} - (B-b); y^{(w)} - 1; a - y^{(w)}; |z| - h/2\right). \tag{3.5}$$

For realistic insect wings, however, we parametrize the wing shape in polar coordinates. As illustrated in Figure 3.2, the shape in the wing plane is described by the center point $\underline{x}_c^{(w)}$, which is arbitrary as long as the function $R(\vartheta)$ is unique for all ϑ. To sample the wing on a computational grid, we need a function $R(\vartheta)$ that can be evaluated for all ϑ. As $R(\vartheta)$ is naturally 2π-periodic, a truncated Fourier series can be used:

$$R(\vartheta) = \frac{a_0}{2} + \sum_{i=1}^{N} a_i \cos(2\pi i\vartheta) + \sum_{i=1}^{N} b_i \sin(2\pi i\vartheta). \tag{3.6}$$

In practice, (3.6) has to be evaluated for all grid points in the vicinity of the wing, which requires $\mathcal{O}(N_x N_y N_z)$ evaluations with a small constant. The computational cost can, however, be significant, which is why (3.6) is evaluated for 25,000 values of ϑ once during initialization. Afterwards, linear interpolation is used for its lower computational cost. The signed distance for such a wing then reads

$$\delta\left(\underline{x}^{(w)}\right) = \max\left(\left|z^{(w)}\right| - h/2; r(\vartheta) - R(\vartheta)\right).$$

If a wing cannot be described by one radius, because $R(\vartheta)$ is not unique, several radii and center points can be used [4].

3.5. Power requirement. Actuating the wings requires power expenditures that are very difficult to measure directly. In numerical simulations, the power can be obtained directly, since the aerodynamic torque moment $\underline{m}$ with respect to the wing pivot point is available from (2.7). The power P_{aero} required to move one wing is found to be

$$P_{\text{aero}} = -\underline{m} \cdot (\underline{\Omega}_w - \underline{\Omega}_b), \tag{3.7}$$

which is equivalent to the definition $P_{\text{aero}} = \int \underline{u} \mathrm{d}\underline{F}$ given in [21]. In addition to the aerodynamic power, inertial power has to be expended, i.e., the power required to

move the wing in a vacuum. As the flapping motion is periodic, its stroke-averaged value is zero. The inertial power P_{inert} is positive if the wing is accelerated (power consumed) and negative if it is decelerated. The definition is

$$P_{\mathrm{inert}} = \underline{\Omega}_w^{(w)} \cdot \left(\underline{\underline{J}}^{(w)} \underline{\dot{\Omega}}_w^{(w)} + \underline{\Omega}_w^{(w)} \times \underline{\underline{J}}^{(w)} \underline{\Omega}_w^{(w)} \right)$$

with the wing tensor of inertia $\underline{\underline{J}}^{(w)}$ [3]. The sum of inertial and aerodynamic power can be negative during deceleration phases, which would mean that the insect can store energy in its muscles. It is unknown to what extent this can be realized, and it is thus often conservatively assumed that energy storage is not possible, in which case the total power P_{total} is given by $P_{\mathrm{total}} = \max\left(P_{\mathrm{inert}} + P_{\mathrm{aero}}, 0\right)$.

3.6. Governing equations in free flight. Until now we have considered the insect to be fixed, i.e., tethered in the computational domain. In free flight, the solid body dynamics equations that describe the motion of the insect have to be solved together with the Navier–Stokes equations that describe the motion of the fluid. The body translation is then governed by

$$M\, \underline{\dot{u}}_{\mathrm{cntr}}^{(g)} = \underline{F}^{(g)},$$

where $\underline{F}^{(g)}$ contains the aerodynamic and gravitational forces and M is the mass of the insect. For simplicity, $\underline{x}_{\mathrm{centr}}$ and $\underline{u}_{\mathrm{centr}}$ correspond to the center of gravity in the case of free flight. To handle the rotational degrees of freedom, we employ a quaternion-based formulation, similar to the one proposed in [20], which avoids the "Gimbal lock" problem. The governing equation for the angular velocity $\underline{\Omega}^{(b)}$ in the body reference frame reads

$$\underline{\underline{J}}^{(b)} \underline{\dot{\Omega}}^{(b)} + \begin{pmatrix} 0 & -\Omega_z^{(b)} & \Omega_y^{(b)} \\ \Omega_z^{(b)} & 0 & -\Omega_x^{(b)} \\ -\Omega_y^{(b)} & \Omega_x^{(b)} & 0 \end{pmatrix} \underline{\underline{J}}^{(b)} \underline{\Omega}^{(b)} = \underline{m}^{(b)},$$

where $\underline{\underline{J}}^{(b)}$ is the moment of inertia around the body axes $\left(x^{(b)}, y^{(b)}, z^{(b)}\right)$ and $\underline{m}$ is the vector of torque moments as defined in (2.7). The skew-symmetric term stems from the change into a moving reference frame. We introduce the normalized quaternion ε with components ε_i, $i = 0, \ldots, 3$, $\sum \varepsilon_i^2 = 1$. The governing equations for the quaternion state are

$$\underline{\dot{\varepsilon}} = \frac{1}{2} \underline{\underline{S}}^T \, \underline{\Omega}^{(b)},$$

with the matrix

$$S = \begin{pmatrix} -\varepsilon_1 & \varepsilon_0 & \varepsilon_3 & -\varepsilon_2 \\ -\varepsilon_2 & -\varepsilon_3 & \varepsilon_0 & \varepsilon_1 \\ -\varepsilon_3 & \varepsilon_2 & -\varepsilon_1 & \varepsilon_0 \end{pmatrix}.$$

Assuming $\underline{\underline{J}}^{(b)}$ to be constant and the body axes to be the principal axes of inertia (i.e., $\underline{\underline{J}}^{(b)}$ is diagonal), the following first order system set of 13 equations is obtained:

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix} x^{(g)}_{\mathrm{cntr}} \\ y^{(g)}_{\mathrm{cntr}} \\ z^{(g)}_{\mathrm{cntr}} \\ u^{(g)}_{\mathrm{cntr,x}} \\ u^{(g)}_{\mathrm{cntr,y}} \\ u^{(g)}_{\mathrm{cntr,z}} \\ \varepsilon_0 \\ \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \Omega^{(b)}_x \\ \Omega^{(b)}_y \\ \Omega^{(b)}_z \end{pmatrix} = \begin{pmatrix} u^{(g)}_{\mathrm{cntr,x}} \\ u^{(g)}_{\mathrm{cntr,y}} \\ u^{(g)}_{\mathrm{cntr,z}} \\ F^{(g)}_x/M \\ F^{(g)}_y/M \\ \left(F^{(g)}_z/M - g\right) \\ \left(-\varepsilon_1\Omega^{(b)}_x - \varepsilon_2\Omega^{(b)}_y - \varepsilon_3\Omega^{(b)}_z\right)/2 \\ \left(\varepsilon_0\Omega^{(b)}_x - \varepsilon_3\Omega^{(b)}_y + \varepsilon_2\Omega^{(b)}_z\right)/2 \\ \left(\varepsilon_3\Omega^{(b)}_x + \varepsilon_0\Omega^{(b)}_y - \varepsilon_1\Omega^{(b)}_z\right)/2 \\ \left(-\varepsilon_2\Omega^{(b)}_x + \varepsilon_1\Omega^{(b)}_y + \varepsilon_0\Omega^{(b)}_z\right)/2 \\ \left(\left(J^{(b)}_y - J^{(b)}_z\right)\Omega^{(b)}_y\Omega^{(b)}_z + m^{(b)}_x\right)/J^{(b)}_x \\ \left(\left(J^{(b)}_z - J^{(b)}_x\right)\Omega^{(b)}_z\Omega^{(b)}_x + m^{(b)}_y\right)/J^{(b)}_y \\ \left(\left(J^{(b)}_x - J^{(b)}_y\right)\Omega^{(b)}_x\Omega^{(b)}_y + m^{(b)}_z\right)/J^{(b)}_z \end{pmatrix}, \tag{3.8}$$

which is solved with the same time discretization as the fluid. The rotation matrix M_{body} is then computed from the quaternion ε_i,

$$M_{body} = \begin{pmatrix} \varepsilon_0^2+\varepsilon_1^2-\varepsilon_2^2-\varepsilon_3^2 & 2\left(\varepsilon_1\varepsilon_2-\varepsilon_3\varepsilon_0\right) & 2\left(\varepsilon_1\varepsilon_3+\varepsilon_2\varepsilon_0\right) \\ 2\left(\varepsilon_1\varepsilon_2+\varepsilon_3\varepsilon_0\right) & \varepsilon_0^2-\varepsilon_1^2+\varepsilon_2^2-\varepsilon_3^2 & 2\left(\varepsilon_2\varepsilon_3-\varepsilon_1\varepsilon_0\right) \\ 2\left(\varepsilon_1\varepsilon_3-\varepsilon_2\varepsilon_0\right) & 2\left(\varepsilon_2\varepsilon_3+\varepsilon_1\varepsilon_0\right) & \varepsilon_0^2-\varepsilon_1^2-\varepsilon_2^2+\varepsilon_3^2 \end{pmatrix}, \tag{3.9}$$

which replaces the definition in (3.1) in the free-flight case. The initial values at time $t = 0$ for ε_i can conveniently be computed from a set of yaw, pitch, and roll angles:

$$\begin{pmatrix} \varepsilon_0 \\ \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \end{pmatrix} = \begin{pmatrix} \cos(\psi/2)\cos(\beta/2)\cos(\gamma/2) + \sin(\psi/2)\sin(\beta/2)\sin(\gamma/2) \\ \sin(\psi/2)\cos(\beta/2)\cos(\gamma/2) - \cos(\psi/2)\sin(\beta/2)\sin(\gamma/2) \\ \cos(\psi/2)\sin(\beta/2)\cos(\gamma/2) + \sin(\psi/2)\cos(\beta/2)\sin(\gamma/2) \\ \cos(\psi/2)\cos(\beta/2)\sin(\gamma/2) - \sin(\psi/2)\sin(\beta/2)\cos(\gamma/2) \end{pmatrix}.$$

In the actual implementation, we multiply the right-hand side of (3.8) with a six component vector with zeros or ones, to deactivate some degrees of freedom.

4. Parallel implementation. Thus far we have described the numerical method employed by the `FluSI` code, which is based on the volume penalization method combined with a pseudospectral discretization. This framework allows for a high degree of modularization, since all geometry is encoded in the mask function and the solid velocity field. The framework is intended to be applicable also for higher Reynolds number flow, for which small spatial and temporal vortical structures appear. To resolve these scales, high resolution and therefore the usage of high-performance computers is required. To this end, our code is designed to compute on massively parallel machines with distributed memory architectures. The parallel implementation is based on the `MPI` protocol and written in `FORTRAN95`.

For the fluid calculations the `P3DFFT` library [26] is used to compute the three-dimensional Fourier transforms. This library provides a parallel data decomposition framework and employs the `FFTW` library [11] for the one-dimensional Fourier transforms. The flow variables are stored on the three-dimensional Cartesian computational grid. Each `MPI` process holds only a portion of the total data, and the parallel decomposition is performed on at most two indices; i.e., a pencil decomposition is used. The

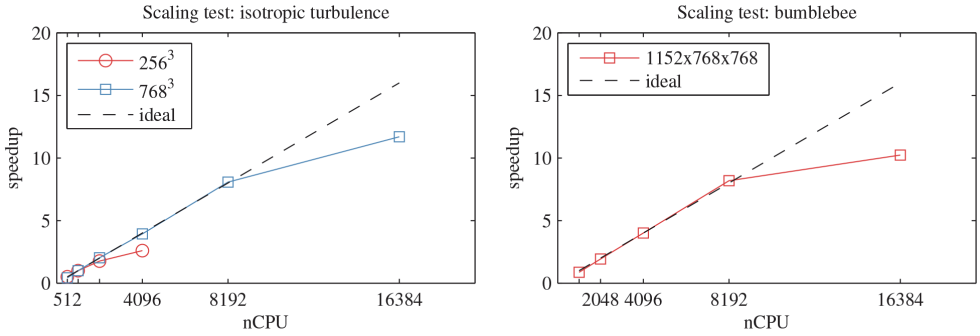

FIG. 4.1. *Parallel scaling tests on a large-scale computing cluster of type IBM BlueGene/Q. Left: Isotropic turbulence simulation without insect, for two different resolutions. Fifty time steps have been performed. The reference simulation is the* 1024 *run. Right: Scaling test for a bumblebee simulation (see section* 6*). Two complete strokes have been computed, and the timing is averaged over the second one. The reference simulation is the* 2048 *cores run.*

x-direction is not split among processes; the code can thus run on $N_y N_z$ processes at most. This limitation is, however, not of practical importance, since $N_y N_z$ usually exceeds the number of available CPUs by orders of magnitude. Besides the Fourier transforms, the operations are local; i.e., they do not require interprocessor communication. The parallel scaling test for the fluid module is shown in Figure 4.1(left) for the case of an isotropic turbulence simulation and two different problem sizes, 256^3 and 768^3. The latter shows good scaling up to 8192 CPU on the IBM BlueGene/Q machine[1] used for the tests.

The insect module is used to generate the χ-function and the solid body velocity field $\underline{u}_s$. It is crucial that the implementation not spoil the favorable scaling properties. The module is therefore designed around a derived insect datatype called `diptera`, which holds all properties of the insect, such as the wing shape, body position, and Fourier coefficients of the wing kinematics. This object is of negligible size, and therefore each `MPI` process holds a copy of it. The task to construct the penalization term is then again completely local. Forces and torques (see (2.6)–(2.7)) are computed efficiently with the `MPI_allreduce` command. It is further necessary to distinguish between distinct parts of the mask function, e.g., body and wings and possibly outer boundary conditions, like channel walls. To accomplish this, we introduce the concept of mask coloring; i.e., we allocate a second `integer*2` array for the mask function that holds the value of the mask color. This allows us to easily exclude, for example, channel walls from the force computation in (2.6), and it limits the memory footprint to two arrays. We further use it to compute the forces acting on the body and wings individually. If the insect's body is tethered, there is no need to reconstruct it every time the mask is updated. The mask coloring can in this case be used easily to delete only the wings from the mask function while keeping the body in memory. This reduces the CPU time requirement of the mask function, since the body occupies a relatively large volume (compared to the wings), and consequently a larger number of grid points are affected.

Figure 4.1(right) shows the scaling test of an insect simulation. The insect model is the bumblebee presented in section 6, but the scaling behavior is the same for any other model. We computed two complete strokes at a resolution of $1152 \times 768 \times 768$,

[1] http://www.idris.fr/eng/turing/

Algorithm 1. General process for a simulation.

1. Initialization. Read parameters from `*.ini` file, allocate memory, set up fluid initial condition, initialize `P3DFFT`
2. Begin loop over time step
   (a) Generate mask function $\chi\left(\underline{x}, t^n\right)$ and solid velocity field $\underline{u}_s\left(\underline{x}, t^n\right)$.
   (b) Determine time step Δt.
   (c) Compute sources $\underline{F}\left(\underline{u}^n\right) = -\underline{\omega}^n \times \underline{u}^n - \frac{\chi^n}{C_\eta}\left(\underline{u}^n - \underline{u}_s^n\right) - \frac{1}{C_{\mathrm{sp}}}\nabla \times \frac{\left(\chi_{\mathrm{sp}}\underline{\omega}^n\right)}{\nabla^2}$.
   (d) Add pressure gradient $\underline{F}^n \leftarrow \underline{F}^n - \nabla\left[\left(\nabla \cdot \underline{F}^n\right)/\nabla^2\right]$.
   (e) Advance fluid to new time level $\underline{u}^{n+1} = \mathrm{AB2}\left(\underline{u}^n, \underline{F}^n\right)$.
   (f) Optional: Advance free-flight equations to new time level, using the AB2 scheme.
   (g) Output: Write flow statistics and flow-field data to disk.
3. When terminal time is reached, free memory and quit.

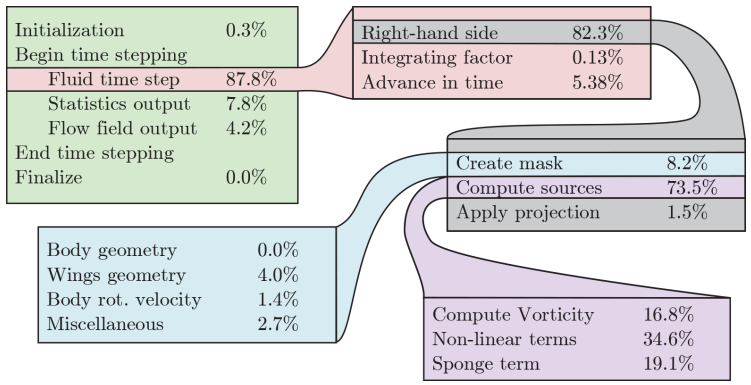

FIG. 4.2. *Partition of computational efforts spent on a bumblebee simulation (see section 6). The resolution is* $1152 \times 768 \times 768$. *Most efforts are spent on advancing the fluid in time, which in turn is largely dominated by the computation of fluid source terms. All percents are with respect to total execution time.*

and the elapsed walltime was measured for the second one. The parallel scaling is virtually the same as in the pure fluid case, indicating that the insect module indeed does not spoil parallel scaling efficiency.

The general process for a simulation is summarized in Algorithm 1. All parameters, like the resolution, etc., are read from a `*.ini` file, which avoids recompiling the code every time a parameter is changed. The code writes time series of quantities like the fluid forces, enstrophy, or aerodynamic power to ASCII `*.t` files. Flow field data, such as the velocity and pressure fields, are stored using the hierarchical data format (`HDF5`) library, in order to guarantee maximum compatibility with third-party applications, such as the open source visualization software `Paraview`, which is used for the visualizations shown in the results section. Figure 4.2 shows how the computing time spent on various tasks is distributed. The computation is dominated by the fluid time stepping (87.8%); the remaining parts are used in the computation of flow statistics and field output. Computing the source terms is by far the most significant contribution, and the generation of the geometry consumes only about 8% of the total time.

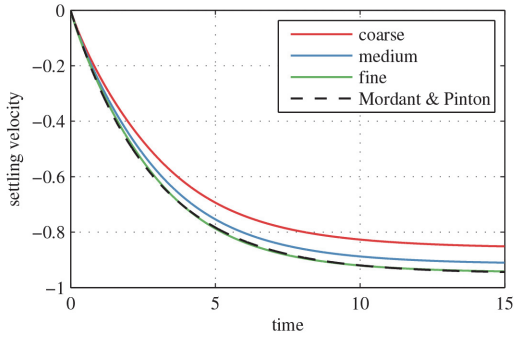

FIG. 5.1. *Settling velocity of a falling sphere, experimental results by Mordant and Pinton* [23]*; we present results for coarse, medium, and fine grids.*

5. Validation tests. For code validation we consider four test cases with increasing complexity: a falling sphere, a rectangular flapping wing, the hovering flight of a fruit fly, and the free flight of a butterfly model.

5.1. Falling sphere. The first test case to validate the flow solver is the sedimentation of a sphere, which in our terminology is an insect without wings and with a spherical body. We consider case 1 proposed by Mordant and Pinton [23], who studied the sedimentation problem experimentally in a water tank. The sphere of unit diameter and mass $M = 1.3404$ is falling in fluid of viscosity $\nu = 0.0228$ and unit density. The dimensionless gravity is $g = 0.8036$, and the terminal settling velocity obtained from the experiments is $U = 0.9488$. We perform a grid convergence test using the domain size $8\times8\times16$ and parameters $N_x\times N_y\times N_z\times C_\eta$ of a coarse $(96\times96\times192\times10^{-3})$, medium $(192 \times 192 \times 384 \times 2.5 \cdot 10^{-4})$, or fine $(384 \times 384 \times 768 \times 6.25 \cdot 10^{-5})$ grid. The number of points per penalization boundary layer is $K = 0.0573$. The results of the convergence study are illustrated in Figure 5.1, and the settling velocity for the finest resolution differs from the experimental findings by less than 1%. The finest resolution required 30 GB of memory and 34,400 CPU hours on 1024 cores to perform 266,667 time steps. The computational cost is relatively high, since small values of C_η are required, as the Reynolds number is small.

5.2. Validation case of a rectangular flapping wing. We consider the setup proposed by Suzuki, Minami, and Inamuro [34, Appendix B2]. It considers only one rectangular wing with a finite thickness, neglecting thus the body and the second wing. The fact that the thickness is finite and the geometry rather simple, compared to actual insects, motivates the choice of this setup. The wing kinematics are given by $\phi = \phi_m \cos(2\pi t)$, $\alpha = \frac{\alpha_m}{\tanh c_\alpha} \tanh(c_\alpha \sin(2\pi t))$, and $\theta = 0$, where $\phi_m = 80°$, $\alpha_m = 45°$, $c_\alpha = 3.3$; the motion is symmetric for the up- and downstroke, and is depicted in Figure 5.2(a)–(b). The rectangular wing and the wing coordinate system are illustrated in Figure 5.2(c). We normalize the distance from pivot to tip to unity, which yields $a = 1.6667$, $b = 0.0667$, $B = 0.4167$ and a wing thickness of $h = 0.04171$. The Reynolds number is set to $\mathrm{Re} = U_{\mathrm{tip}}B/\nu = 100$ with $U_{\mathrm{tip}} = 2\pi\phi_m$, yielding the kinematic viscosity $\nu = 0.0366$. In the present simulations, we discretize the domain of size $3 \times 3 \times 3$ using $512 \times 512 \times 512$ points and a penalization parameter of $C_\eta = 1.25 \cdot 10^{-4}$ ($K = 0.365$). The reference computation in [34] is performed in a domain of size $4.16 \times 4.16 \times 4.16$, using a fine grid near the wing and a coarse one in the far-field. Based on the resolution of the fine grid, $\Delta x = B/50$, the corresponding

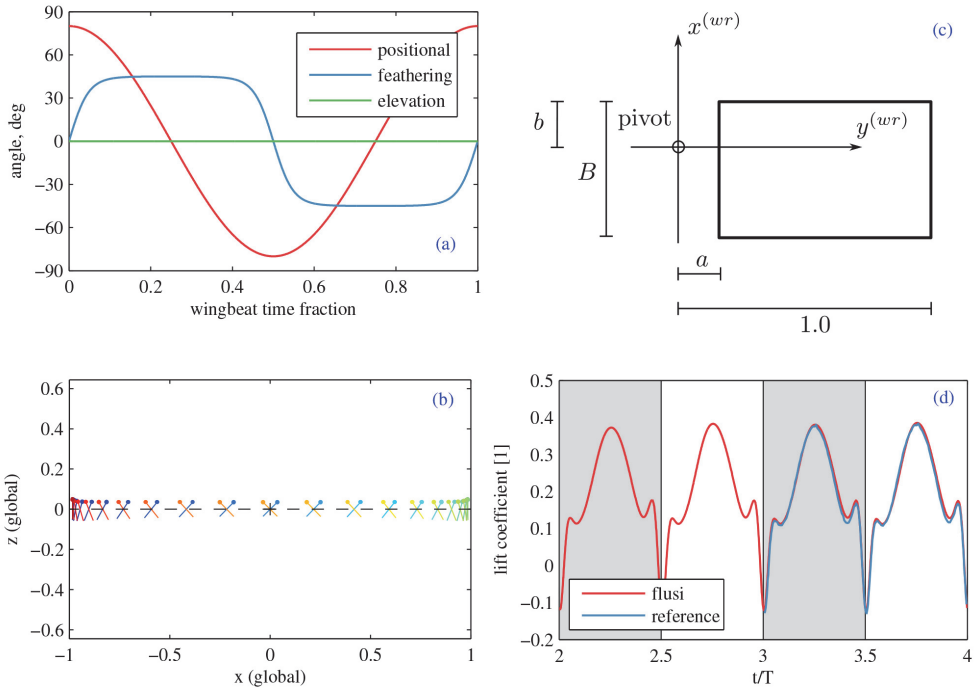

FIG. 5.2. *Flapping rectangular wing.* (a) *Kinematics used in the test case, as given by Suzuki, Minami, and Inamuro* [34]. (b) *Visualization of the wing kinematics by a chord section (without body; color represents time).* (c) *Geometry of the flapping rectangular wing. In contrast to* [34]*, we normalize the distance between pivot and wing tip to unity.* (d) *Time evolution of the vertical force acting on the wing for the last two strokes, with the reference solution presented in* [34].

equidistant resolution would be 500^3. In our simulations, the body reference point is located at $\underline{x}^{(g)}_{\mathrm{cntr}} = (1.5, 1.5, 1.7)$ and coincides with the wing pivot point; thus $\underline{x}^{(b)}_{\mathrm{pivot}} = 0$. The orientation of the body coordinate system is given by $\psi = 0°$, $\beta = -45°$, $\gamma = 45°$, and $\eta = -45°$, where the 45° yaw angle was added to keep the wing as far away from the vorticity sponges as possible. A total of four strokes has been computed, starting with a quiescent initial condition, $\underline{u}\,(\underline{x}, t = 0) = 0$. The outer boundary conditions on the domain are homogeneous Dirichlet conditions in the z-direction and a vorticity sponge, extending over 32 grid points with $C_{\mathrm{sp}} = 10^{-1}$, in the remaining directions. The simulation required 35 GB of memory and 5785 CPU hours on 1024 cores. A total of 27,701 time steps was performed.

The resulting time series of the vertical force is illustrated in Figure 5.2(d). It takes the first two wingbeats to develop a periodic state, since the motion is impulsively started, and then the following strokes are almost identical. The present solution agrees well with the reference solution, and the relative root-mean-squared difference is $\|F - F_{\mathrm{ref}}\|_2 \,/\, \|F_{\mathrm{ref}}\|_2 \approx 4\%$ over the last two periods.

5.3. Hovering flight of a fruit fly model. The third validation test presents a comparison of aerodynamic measures of a hovering fruit fly with the results reported by Maeda and Liu [21]. Their simulations are based on overset grids; i.e., a body-fitted grid for the wings ($65 \times 65 \times 11$ each) and the body ($61 \times 61 \times 9$), as well as a background grid ($161 \times 141 \times 127$), have been used to solve the incompressible

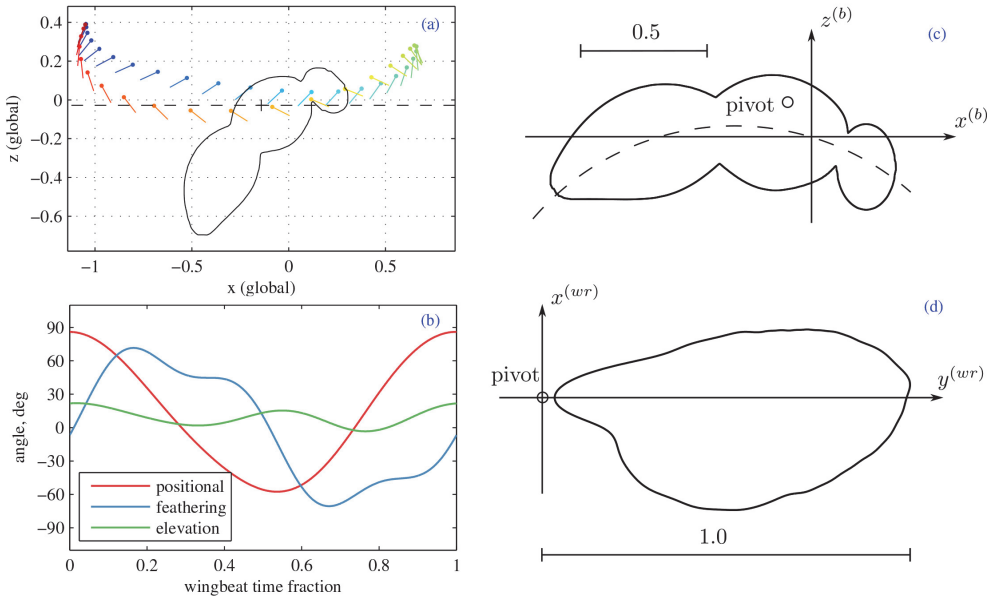

FIG. 5.3. *Hovering fruit fly, for comparison with* [21]. (a) *Wingbeat kinematics in the side view.* (b) *Time evolution of the wing angles.* (c) *Drawing of the insect's body in the plane* $y^{(b)} = 0$. *The body is rotationally symmetric with a circular center line.* (d) *Drawing of the wing shape, together with its pivot point.*

Navier–Stokes equations, approximated in the artificial compressibility formulation, using a finite volume discretization.

The fruit fly considered is defined in Figure 5.3. The wing length from pivot point to wing tip, $R = 2.47\,\mathrm{mm}$, the fluid density $\varrho_f = 1.225\,\mathrm{kg/m^3}$, and the wingbeat frequency $f = 218\,\mathrm{Hz}$ are used for normalization. The body, depicted in Figure 5.3(a),(c), has an elliptical cross section with center points following an arched centerline of radius $r_{\mathrm{bc}} = 0.94644$ centered at $x_{\mathrm{bc}}^{(b)} = -0.244769$, $z_{\mathrm{bc}}^{(b)} = -0.9301256$. The wing pivot points are located at $\underline{x}_{\mathrm{pivot,rl}}^{(b)} = (-0.12, \pm 0.1445, 0.08)$. To facilitate reproduction, the supplementary material for this paper contains `STL` files (StereoLithography) of the fruitfly geometry, as well a parameter file that can be used to reproduce the results with `FluSI`. The insect is tethered at $\underline{x}_c^{(g)} = (1.6, 1.6, 1.9)$ in a computational domain of size $3.2 \times 3.2 \times 3.2$, discretized with $640 \times 640 \times 640$ Fourier modes and a penalization parameter of $C_\eta = 1.15 \cdot 10^{-4}$ ($K = 0.23$). Its body orientation is given by $\psi = 0$, $\beta = -45°$, $\gamma = 45°$, and $\eta = -45°$. The fruit fly hovers; thus the body position and orientation do not change in time. A total of four wing beats has been computed, and the initial condition is fluid at rest, $\underline{u}\,(\underline{x}, t = 0) = \underline{0}$. We apply a vorticity sponge in the x- and y-directions (32 grid points thick with $C_{\mathrm{sp}} = 10^{-1}$) and impose no-slip boundary conditions in the z-direction; i.e., we impose a floor and ceiling. The simulation required 68 GB of memory and 15,100 CPU hours on 1024 cores. A total of 48,200 time steps was performed.

The wing shape is illustrated in Figure 5.3(d). It has a mean chord $c_m = A/R = 0.33R = 0.82$ mm. The Reynolds number is $\mathrm{Re} = U_{\mathrm{tip}} c_m/\nu = 2\Phi R f c_m/\nu = 142$, where $\Phi = 2.44\,\mathrm{rad}$ is the stroke amplitude of the positional angle, U_{tip} is the mean wingtip velocity, and $\nu = 1.5{\cdot}10^{-5}\,\mathrm{m^2/s}$ is the kinematic viscosity of air. The time evolution of the wing kinematics is illustrated in Figure 5.3(b). The up- and downstroke

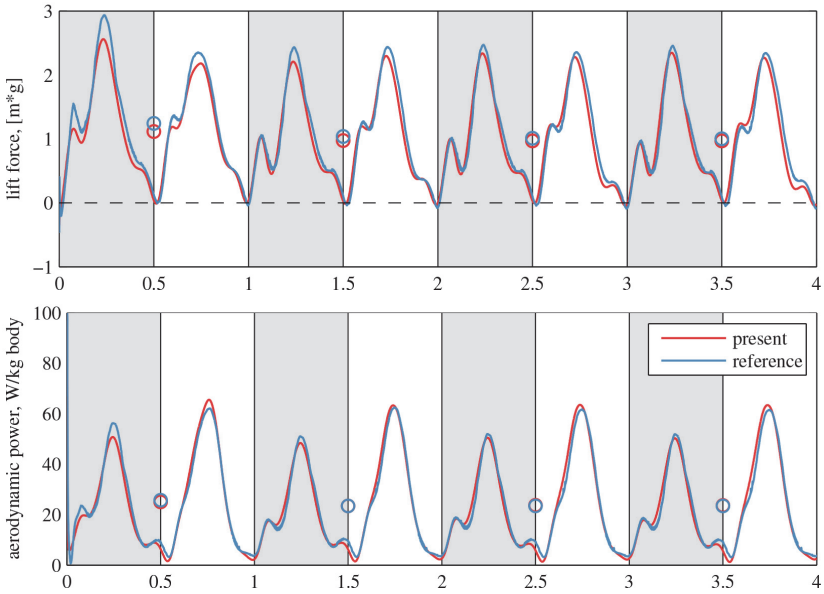

FIG. 5.4. *Hovering fruit fly, comparison with* [21]. *Top: Total vertical force. Bottom: Aerodynamic power. Gray shaded areas denote upstrokes; stroke averages are indicated by circles. The mean values during the last stroke differ by* 3.26% *and* 1.00% *for the vertical force and the power, respectively.*

are not symmetric, and the wing trajectory (Figure 5.3(a)) shows the characteristic ∪-shape.

The results for the lift force, normalized with the weight, and the aerodynamic power in W/kg body mass are presented in Figure 5.4. As the mass of the model insect was not given in [21], we assumed the fourth stroke of the reference data to balance the weight, yielding $m = 1.02\,\mathrm{mg}$. Small circles at midstroke indicate stroke-averaged values. For the stroke-averaged vertical force, we find respectively 10.03%, 6.19%, 4.03%, and 3.26% relative difference from the reference data for the four strokes computed, which can be explained by the impulsively started motion at the beginning of the first stroke. The time evolution, e.g., the occurrence of peaks, is very similar in both data sets. The agreement is even better for the aerodynamic power, with relative differences of 0.57%, 0.06%, −0.95%, and −1.00% for the stroke-averaged values. Both power and lift peak during the translation phase of the up- and downstroke and reach a minimum value at the reversals.

The flow field generated by the fruit fly model is visualized in Figure 5.5 by isosurfaces of the Q-criterion, which can, for incompressible flows, be computed as $Q = \nabla^2 p/2$; see [18, p. 23]. The flow field exhibits the typical features, such as a leading edge vortex and a wingtip vortex.

5.4. Butterfly model in free flight. To complete the validation section, we consider the freely flying butterfly model presented in [34, section 5.3]. The body consists of a thin rod with quadratic wings of length $R = 1$, and the wing kinematics is inspired by a butterfly. Figure 5.6(top row) illustrates the prescribed wingbeat kinematics, converted to the convention used in this work. The Reynolds number is $Re = 300 = U_{\mathrm{tip}} R/\nu$, where $U_{\mathrm{tip}} = \pi$ is the mean wingtip velocity. The mass of the butterfly is $M = 38$, and the moments of inertia of the body are given by

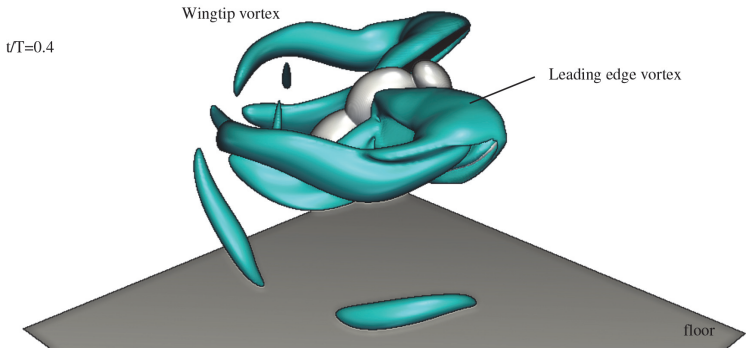

FIG. 5.5. *Snapshot of the vortical structures generated by a hovering fruit fly model, visualized by isosurfaces of the Q-criterion ($Q = 100$) shortly before the ventral stroke reversal $t/T = 0.4$. Typical feature of flapping flight, such as the leading edge and wingtip vortex, are visible.*

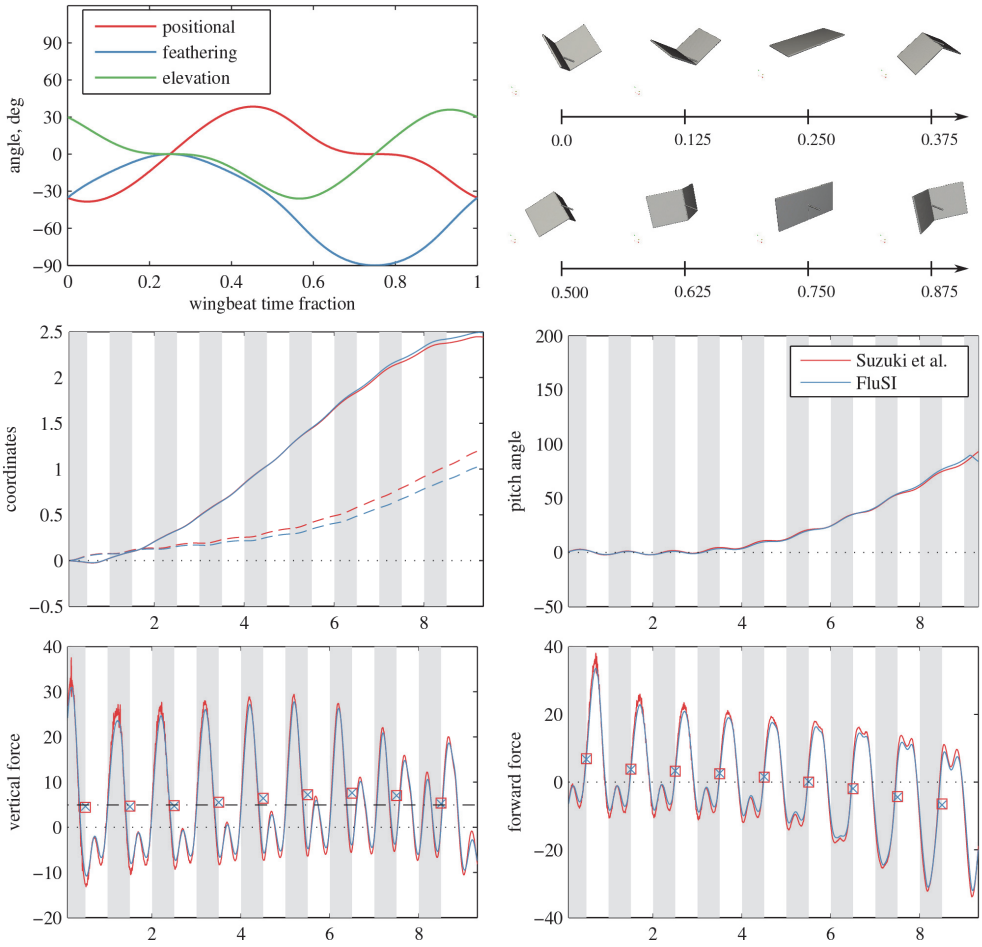

FIG. 5.6. *Freely flying butterfly model for comparison with* [34]. *Top: Kinematics and illustration of the wingbeat. Middle left: Coordinates of the center of mass for the horizontal (solid lines) and vertical (dashed line) component. Middle right: Body pitch angle β. Bottom: Vertical (left) and forward (right) force component. Symbols indicate stroke averages. Dash-dotted lines mark weight. Overall agreement is good.*

$J_y^{(b)} = J_z^{(b)} = 3.1667$. The roll motion is inhibited (five degrees of freedom), and the mass of the wings is neglected. Gravitational acceleration is given by $g = 0.1304$. Present computations are performed on a domain of size $6 \times 6 \times 6$ using a resolution of $1024 \times 1024 \times 1024$, and the penalization constant is $C_\eta = 1.4 \cdot 10^{-4}$ ($K = 0.2$). We apply a vorticity sponge in all directions, centered around the moving center of the insect $\underline{x}_{\mathrm{cntr}}^{(g)}$. We computed nine strokes (65,066 time steps); the computation allocated 200 GB of memory and consumed 79,915 CPU hours on 4096 cores. The parameter files used in this computation are in the supplementary materials.

Figure 5.6(middle row) compares the coordinates of the center of mass and the body pitch angle with the reference solution, showing good agreement with the reference computation. A slight deviation in the vertical component can be observed, which can be explained by the fact that the slight differences in the forces (see Figure 5.6(bottom row)) are integrated twice with respect to time. The general agreement is good, and we conclude that our code is validated.

6. Application to a bumblebee model in forward flight. In order to demonstrate the applicability of the software environment to more complex insects at higher Reynolds number, we consider in this section a different insect model with a Reynolds number of $\mathrm{Re} = 2\Phi R f c_m/\nu = 2042$. The model is derived from a bumblebee, and its detailed morphology and kinematics can be found in the supplementary materials (`STL` file of the geometry as well as the parameter files that can be used to reproduce the results with `FluSI`). The model has also been used in [10, supplemental materials]. The body contains details such as the legs and antennae, since they can be easily included in the present framework. The bumblebee is considered in forward flight at $u_\infty = 1.246$. The domain size is set to $6 \times 4 \times 4$ and discretized with $1152 \times 768 \times 768$ grid points. The penalization parameter is set to $C_\eta = 2.5 \cdot 10^{-4}$ ($K = 0.074$). Four strokes have been simulated, requiring 26,467 CPU hours on 1024 cores and 153.6 GB of memory.

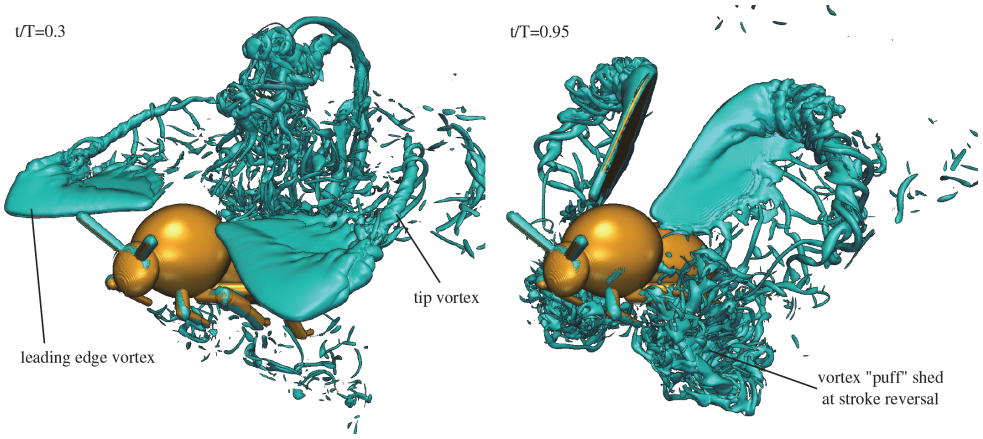

FIG. 6.1. *Flow field generated by the model bumblebee in forward flight, visualized by the* $\|\underline{\omega}\| = 100$ *isosurface of vorticity magnitude.*

Figure 6.1 visualizes the flow field generated by the model by the $\|\underline{\omega}\| = 100$ isosurface of vorticity magnitude, for the times $t/T = 0.3$ and 0.95, where T is the period time. At $t/T = 0.3$, the leading edge vortex and the wingtip vortex are clearly visible, yet the flow field presents much smaller scales than in the case of

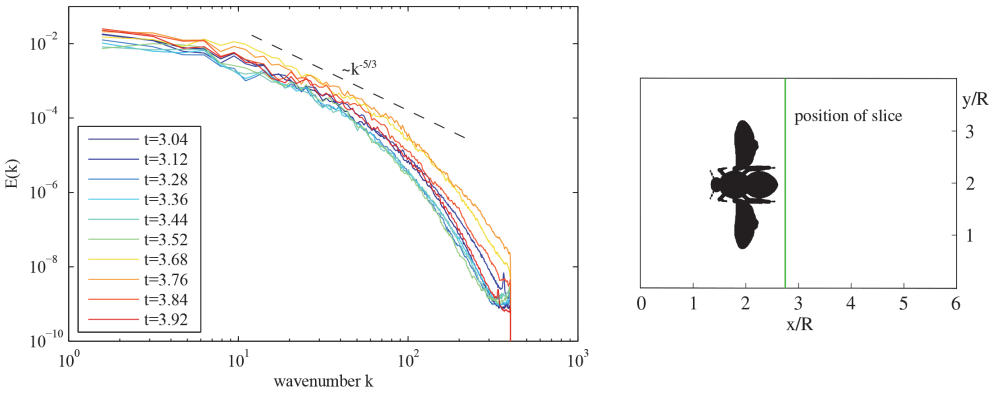

FIG. 6.2. *Left: Radial energy spectra in a slice perpendicular to the flow direction ($x = 2.67$). The spectra are full and exhibit an inertial and dissipative range. The resolution is $1152 \times 768 \times 768$; thus the highest resolved wavenumber is $k_{\max} = 256 \cdot (2\pi/4)$, due to the dealiasing using the 2/3 rule. Right: Position of the slice and insect in the computational domain of size $6 \times 4 \times 4$.*

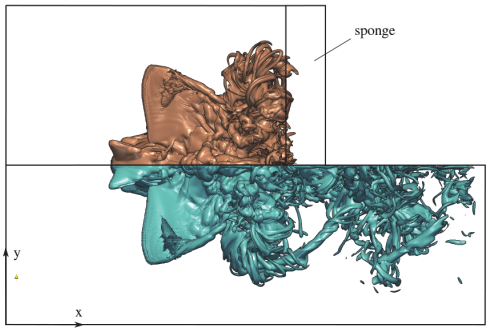

FIG. 6.3. *Influence of the vorticity sponge in a bumblebee simulation. Shown is the $\|\underline{\omega}\| = 20$ isosurface of vorticity magnitude. A simulation with a longer ($\ell_x = 6$, blue) and a shorter ($\ell_x = 4$, orange) domain has been performed. In the shorter simulation, the sponge occupies the region $3.5 \le x \le 4$. The sponge constant is $C_{\rm sp} = 10^{-1}$.*

a fruitfly (Figure 5.5). At $t/T = 0.95$, which is shortly before the beginning of a new cycle, the vortex "puff" shed at the stroke reversal ($t/T \approx 0.5$) is visible. The flow field is both spatially and temporally intermittent. Figure 6.1 gives the impression of a turbulent flow field, which can be quantified by the energy spectra of the velocity in a two-dimensional slice perpendicular to the flow direction. Figure 6.2 shows the chosen position of the slice at $x = 2.67$, and the radial energy spectra for ten times throughout the cycle. At any time t/T, the energy spectrum $E(k) = 1/2 \sum_{k-1/2 \le |\underline{k}| < k+1/2} |\widehat{u}_x(\underline{k})|^2 + |\widehat{u}_z(\underline{k})|^2 + |\widehat{u}_z(\underline{k})|^2$, where $\underline{k} = (k_x, k_y)$, is full and exhibits an inertial range with a $k^{-5/3}$ slope, which is a typical feature of turbulence.

As described in section 2.3, a vorticity sponge technique has been used to overcome the periodicity. Let us briefly discuss the influence of the sponge and the choice of parameters. Figure 6.3 visualizes the flow field from two simulations, one with a shorter domain ($4 \times 4 \times 4$) and the previously discussed one. In the shorter domain, a sponge is active in the region $3.5 \le x \le 4$ with a constant of $C_{\rm sp} = 10^{-1}$.

7. Conclusions and perspectives. This article presents the open source software `FluSI` (https://github.com/pseudospectators/FLUSI) for the numerical simula-

tion of the aerodynamics of flapping insect flight running on massive parallel computer architectures. Different benchmarks demonstrated the efficiency of the code and showed its validity in comparison with results from the literature. The numerical method is based on a Fourier pseudospectral discretization of the three-dimensional incompressible Navier–Stokes equations. Thus no artificial numerical diffusion or dispersion is introduced into the discretization. The no-slip boundary conditions for the complexly shaped and time varying geometry of the flapping wings and the insect body are imposed with the volume penalization method. The penalization parameter is chosen such that the modeling error, due to penalization, and the discretization error are balanced. The computational cost of the flow solver is essentially due to the Fourier transforms. Benefiting from the efficient implementation of three-dimensional FFT, excellent scaling on large-scale computing clusters has thus been obtained. A limitation of the current approach is that equidistant Cartesian grids are required and a compromise between the domain size and the number of grid points has to be imposed. The modular structure of the `FluSI` code permits us to design different and complex geometries for the insect shape and its wings easily and also to change their kinematics. For the free flight option, equations in a quaternion-based formulation are solved. Different flow configurations, like channel flows with laminar or turbulent inflow or including bluff bodies of almost arbitrary shape, are possible using penalization together with a sponge technique.

Acknowledgments. This work was granted access to the HPC resources of Aix-Marseille Université financed by the project equip@meso (ANR-10-EQPX-29-01) of the program "investissements d'avenir" supervised by the Agence Nationale pour la Recherche, as well as to the HPC resources of IDRIS (Institut du Développement et des Ressources en Informatique Scientifique) under project number i20152a1664. The authors also wish to thank Masateru Maeda and Hao Liu for fruitful discussions on insect modeling, and Malcolm Roberts for contributing to the code development.

REFERENCES

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

[2] 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), pp. 1–4.

[3] G. J. Berman and Z. J. Wang, *Energy-minimizing kinematics in hovering insect flight*, J. Fluid Mech., 582 (2007), pp. 153–168.

[4] G. Bimbard, D. Kolomenskiy, O. Bouteleux, J. Casas, and R. Godoy-Diana, *Force balance in the take-off of a pierid butterfly: Relative importance and timing of leg impulsion and aerodynamic forces*, J. Exp. Biol., 216 (2013), pp. 3551–3563.

[5] C. Canuto, M. Y. Hussaini, A. Quarteroni, and T. Zang, *Spectral Methods in Fluid Dynamics*, Springer-Verlag, Berlin, 1986.

[6] G. Carbou and P. Fabrie, *Boundary layer for a penalization method for viscous incompressible flow*, Adv. Differential Equations, 8 (2003), pp. 1453–1480.

[7] J. W. Cooley and J. W. Tukey, *An algorithm for the machine calculation of complex Fourier series*, Math. Comp., 19 (1965), pp. 297–301.

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

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

[10] T. Engels, D. Kolomenskiy, K. Schneider, J. Sesterhenn, and F.-O. Lehmann, *Bumblebee flight in heavy turbulence*, Phys. Rev. Lett., 116 (2016), 028013.

[11] M. Frigo and S. G. Johnson, *The design and implementation of FFTW3*, Proc. IEEE, 94 (2005), pp. 216–231.
[12] M. Hejlesen, P. Koumoutsakos, A. Leonard, and J. Walther, *Iterative Brinkman penalization for remeshed vortex methods*, J. Comput. Phys., 280 (2015), pp. 547–562.
[13] C. Introini, M. Belliard, and C. Fournier, *A second order penalized direct forcing for hybrid Cartesian/immersed boundary flow simulations*, Comput. & Fluids, 90 (2014), pp. 21–41.
[14] C. Ji, A. Munjiza, and J. Williams, *A novel iterative direct-forcing immersed boundary method and its finite volume applications*, J. Comput. Phys., 231 (2012), pp. 1797–1821.
[15] B. Kadoch, D. Kolomenskiy, P. Angot, and K. Schneider, *A volume penalization method for incompressible flows and scalar advection-diffusion with moving obstacles*, J. Comput. Phys., 231 (2012), pp. 4365–4383.
[16] J. Kim and P. Moin, *Application of a fractional-step method to incompressible Navier-Stokes equations*, J. Comput. Phys., 59 (1985), pp. 308–323.
[17] 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 (2009), pp. 5687–5709.
[18] M. Lesieur, O. Métais, and P. Comte, *Large-Eddy Simulations of Turbulence*, Cambridge University Press, Cambridge, UK, 2005.
[19] H. Liu, *Integrated modeling of insect flight: From morphology, kinematics to aerodynamics*, J. Comput. Phys., 228 (2009), pp. 439–459.
[20] M. Maeda, N. Gao, N. Nishihashi, and H. Liu, *A free-flight simulation of insect flapping flight*, J. Aero Aqua Bio-mech., 1 (2010), pp. 71–79.
[21] M. Maeda and H. Liu, *Ground effect in fruit fly hovering: A three-dimensional computational study*, J. Biomech. Sc. Engin., 8 (2013), pp. 344–355.
[22] R. Mittal and G. Iaccarino, *Immersed boundary methods*, in Annual Review of Fluid Mechanics, Annu. Rev. Fluid Mech. 37, Annual Reviews, Palo Alto, CA, 2005, pp. 239–261.
[23] N. Mordant and J.-F. Pinton, *Velocity measurement of a settling sphere*, Eur. Phys. J. B, 18 (2000), pp. 343–352.
[24] 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 (2014), pp. 301–338.
[25] S. Osher and R. Fedkiw, *Level Set Methods and Dynamic Implicit Surfaces*, Springer, New York, 2003.
[26] D. Pekurovsky, *P3DFFT: A framework for parallel computations of Fourier transforms in three dimensions*, SIAM J. Sci. Comput., 34 (2012), pp. C192–C209.
[27] C. S. Peskin, *The immersed boundary method*, Acta Numer., 11 (2002), pp. 479–517.
[28] R. Peyret, *Spectral Methods for Incompressible Viscous Flow*, Springer, Berlin, Heidelberg, 2002.
[29] R. Ramamurti and W. Sandberg, *A computational investigation of the three-dimensional unsteady aerodynamics of Drosophila hovering and maneuvering*, J. Exp. Biol., 210 (2009), pp. 881–896.
[30] P. Schlatter, N. Adams, and L. Kleiser, *A windowing method for periodic inflow/outflow boundary treatment of non-periodic flows*, J. Comput. Phys., 206 (2005), pp. 505–535.
[31] K. Schneider, *Numerical simulation of the transient flow behaviour in chemical reactors using a penalisation method*, Comput. & Fluids, 34 (2005), pp. 1223–1238.
[32] K. Schneider, *Immersed boundary methods for numerical simulation of confined fluid and plasma turbulence in complex geometries: A review*, J. Plasma Phys., 81 (2015), 435810601.
[33] K. Schneider and M. Farge, *Numerical simulation of the transient flow behaviour in tube bundles using a volume penalization method*, J. Fluids Struct., 20 (2005), pp. 555–566.
[34] 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 (2015), pp. 659–695.
[35] J. F. Thompson, Z. Warsi, and C. W. Mastin, *Numerical Grid Generation: Foundations and Applications*, North–Holland, Amsterdam, 1985.
[36] M. Uhlmann, *An immersed boundary method with direct forcing for the simulation of particulate flows*, J. Comput. Phys., 209 (2005), pp. 448–476.