Kenji Takizawa · Tayfun E. Tezduyar

Main Aspects of the Space–Time Computational FSI Techniques and Examples of Challenging Problems Solved

Abstract Flow problems with moving boundaries and interfaces include fluid–structure interaction (FSI) and a number of other classes of problems, have an important place in engineering analysis and design, and offer some formidable computational challenges. Bringing solution and analysis to such flow problems motivated the development of the Deforming-Spatial-Domain/Stabilized Space–Time (DSD/SST) method. Since its inception, the DSD/SST method and its improved versions have been applied to a diverse set of challenging problems with a common core computational technology need. The classes of problems solved include free-surface and two-fluid flows, fluid–object and fluid–particle interaction, FSI, and flows with solid surfaces in fast, linear or rotational relative motion. Some of the most challenging FSI problems, including parachute FSI and arterial FSI, are being solved and analyzed with the DSD/SST method as a core technology. Better accuracy and improved turbulence modeling were brought with the recently-introduced variational multiscale (VMS) version of the DSD/SST method, which is called DSD/SST-VMST (also ST-VMS). In specific classes of problems, such as parachute FSI, arterial FSI, aerodynamics of flapping wings, and wind-turbine aerodynamics, the scope and accuracy of the FSI modeling were increased with the special ST FSI techniques targeting each of those classes of problems. This article provides an overview of the core ST FSI technique, its recent versions, and the special ST FSI techniques. It also provides examples of challenging problems solved and analyzed in parachute FSI, arterial FSI, aerodynamics of flapping wings, and wind-turbine aerodynamics.

Keywords Parachute · Artery · Flapping wings · MAV · Wind turbine · FSI · Space–time · VMS

1 Introduction

Flows with moving boundaries and interfaces include fluid–structure interaction (FSI), fluid–object interaction (FOI), fluid–particle interaction (FPI), free-surface and multi-fluid flows, and flows with solid surfaces in fast, linear or rotational relative motion. These problems are frequently encountered in engineering analysis and design, pose some of the most formidable computational challenges, and have a common core computational technology need. That crucial need motivated the development of the Deforming-Spatial-Domain/Stabilized Space–Time (DSD/SST) method [1; 2; 3; 4; 5; 6; 7; 8], which is a general-purpose interface-tracking (moving-mesh) technique, as a core computational technology. The DSD/SST method is an alternative to the Arbitrary Lagrangian–Eulerian (ALE) finite element formulation [9], which is the most widely used moving-mesh technique, with increased emphasis on FSI in recent years (see, for example, [10; 11; 12; 13; 14; 15; 16; 17; 18; 19; 20; 21; 22; 23; 24; 25; 26; 27; 28; 29; 30; 31; 32; 33; 34; 35; 36; 37; 38; 39; 40; 41; 42]). Though less widely used than the ALE formulation, over the past 20 years the DSD/SST method has been applied to some of the most challenging moving-interface problems. The classes of problems solved with the DSD/SST method since its inception include the free-surface and multi-fluid flows [1; 3; 43; 44; 45; 46], FOI [1; 2; 3; 47; 44; 48; 46], aerodynamics of flapping wings [49; 50; 51; 52; 53], flows with surfaces in fast, linear or rotational relative motion [44; 45; 54; 55; 28; 56], compressible flows [44], shallow-water flows [45; 57], FPI [44; 58; 59; 60; 45], and FSI [47; 61; 49; 62; 63; 64; 65; 66; 67; 68; 5; 69; 70; 71; 72; 73; 74; 75; 76; 77; 78; 79; 80; 81; 82; 83; 84; 85; 86; 87; 88; 89; 90; 91; 92; 6; 93; 94; 95; 96; 97; 98; 7; 99; 100; 101; 102].

In the DSD/SST formulation, as it was originally envisioned, the ST computations are carried out one ST “slab” at a time, where the “slab” is the slice of the ST domain between the time levels n and n + 1. The basis functions are continuous within a ST slab, but discontinuous from one ST slab to another. The original
DSD/SST method [1] is based on the SUPG/PSPG stabilization, where “SUPG” and “PSPG” stand for the Streamline-Upwind/Petrov-Galerkin [103] and Pressure-Stabilizing/Petrov-Galerkin [1; 104] methods. Starting in its very early years, the DSD/SST method also included the “LSC” (least-squares on incompressibility constraint) stabilization. New versions of the DSD/SST method have been introduced since its inception, including those in [5], which have been serving as the core numerical technology in the majority of the ST FSI computations carried out in recent years. The most recent DSD/SST method is the ST version [6; 7] of the residual-based variational multiscale (VMS) method [105; 106; 107; 108]. It was named “DSD/SST-VMST” (i.e. the version with the VMS turbulence model) in [6], which was also called “ST-VMS” in [7]. The original DSD/SST method was named “DSD/SST-SUPS” in [6] (i.e. the version with the SUPG/PSPG stabilization), which was also called “ST-SUPS” in [8].

Moving-mesh methods require mesh update methods. Mesh update typically consists of moving the mesh for as long as possible and remeshing as needed. With the key objectives being to maintain the element quality near solid surfaces and to minimize frequency of remeshing, a number of advanced mesh update methods [109; 43; 110; 45; 5] were developed to be used with the DSD/SST method, including those that minimize the deformation of the layers of small elements placed near solid surfaces.

An ST method will naturally involve more computational cost per time step than an ALE method, but it gives us the option of using higher-order basis functions in time, including the NURBS basis functions, which have been used very effectively as spatial basis functions (see [111; 12; 16; 112]). This of course increases the order of accuracy in the computations [89; 6; 7], and the desired accuracy can be attained with larger time steps, but there are positive consequences beyond that. The ST context provides us better accuracy and efficiency in temporal representation of the motion and deformation of the moving interfaces and volume meshes, and better efficiency in remeshing. This has been demonstrated in a number of 3D computations, specifically, flapping-wing aerodynamics [50; 51; 52; 53], separation aerodynamics of spacecraft [100], and wind-turbine aerodynamics [56].

There are some advantages in using a discontinuous temporal representation in ST computations. For a given order of temporal representation, we can reach a higher order accuracy than one would reach with a continuous representation of the same order. When we need to change the spatial discretization (i.e. remesh) between two ST slabs, the temporal discontinuity between the slabs provides a natural framework for that change. There are advantages also in continuous temporal representation. We obtain a smooth solution, NURBS-based when needed. We also can deal with the computed data in a more efficient way, because we can represent the data with fewer temporal control points, and that reduces the computer storage cost. These advantages motivated the development of the ST computation techniques with continuous temporal representation (ST-C) [113].

The core and special ST FSI methods mentioned above were essentially all motivated by the need for the solution and analysis of specific classes of challenging problems, such as parachute FSI, arterial FSI, aerodynamics of flapping wings, and wind-turbine aerodynamics. This can be seen from the ST articles cited in the first paragraph, especially the articles since 2008, and will also be seen from the examples we will present in this paper. In the case of the parachute FSI, the special methods were motivated also by the need for supporting the design process for the NASA spacecraft parachutes.

An overview of the core and special methods will be provided in Sections 2 and 3. Examples of the challenging problems solved will be presented in Section 4, and the concluding remarks will be given in Section 5.

### 2 Core methods

The DSD/SST method, given in its DSD/SST-SUPS (or ST-SUPS) form by the the equation below,

\[
\int_{Q_n} w^h \cdot \rho \left( \frac{\partial u^h}{\partial t} + u^h \cdot \nabla u^h - f^h \right) dQ \\
+ \int_{Q_n} \varepsilon(u^h) : \sigma(u^h, p^h) dQ - \int_{(P_n)_h} \nabla \cdot h^d dP \\
+ \int_{Q_n} q^h \nabla \cdot u^h dQ + \int_{\Omega_n} (w^h)_n \cdot \rho \left( (u^h)_n - (u^h)_n \right) d\Omega \\
+ \sum_{c=1}^{(n_{el})_n} \int_{Q^c_{n}} \frac{\mu_{SUPG}}{\rho} \left[ \frac{\partial w^h}{\partial t} + u^h \cdot \nabla w^h \right] \\
\quad + \tau_{PSPG} \nabla q^h \cdot \mathbf{r}_M(u^h, p^h) dQ \\
+ \sum_{c=1}^{(n_{el})_n} \int_{Q^c_{n}} \mu_{LSIC} \nabla \cdot w^h \rho C(u^h) dQ = 0. \\
\tag{1}
\]

has all the ingredients of the semi-discrete SUPG/PSPG finite element formulation. That includes the test functions, domain integrations, stress terms that have been integrated by parts, boundary integrations, and the SUPG, PSPG and LSIC stabilization terms with stabilization parameters \( \tau_{SUPG} \), \( \tau_{PSPG} \) and \( \nu_{LSIC} \) (see [5] for the definitions). The stabilization is residual based, where the residuals of the momentum equation and incompressibility constraint, \( \mathbf{r}_M(u^h, p^h) \) and \( \mathbf{r}_C(u^h) \), appear as factors in the stabilization terms.

In an ST formulation, the domain and boundary integrations are those associated with the ST slab (see Figure 1). The velocity and pressure \((u^h \text{ and } p^h)\) and the corresponding test functions \((w^h \text{ and } q^h)\) are continuous in space but discontinuous from one ST slab to another. While this increases the number of unknowns per
grid point and the computational cost per time step, it also increases the accuracy of the formulation as can be clearly seen in Figure 2. We note that the version of the

dSD/SST formulation that uses lower-order functions in time would be comparable to an ALE method in terms of the computational cost and algorithmic nature.

In addition, an ST method gives us the option of using higher-order functions in time in an ST slab (see Figure 3). Using higher-order functions in temporal representation of the motion and deformation of the interfaces gives us better accuracy. In addition, using higher-order temporal functions in mesh motion and remeshing gives us a more efficient way of managing the mesh update.

Because of the moving-mesh nature of the DSD/SST method, the higher mesh resolution near fluid–solid interfaces can follow the interface, yielding a higher accuracy in resolving the boundary layers near solid surfaces. This concept is illustrated with the simple example in Figure 4. On the other hand, in a typical interface-capturing (nonmoving-mesh) method, there is no higher-resolution mesh near fluid–solid interfaces to begin with. This is illustrated in the hypothetical case shown in Figure 5. Even if one is willing to pay the price of using a very high-resolution mesh everywhere in the domain and that resolution is sufficient for some spheres, it may not be sufficient for others.
Fig. 5: In a nonmoving-mesh method, there is no higher-resolution mesh near fluid-solid interfaces. Even if the mesh resolution everywhere is sufficient for some spheres, it may not be sufficient for others.

The conservative form of DSD/SST-VMST (or ST-VMS) method is given as

\[
\begin{align*}
\int_{Q_n} & w^h \cdot \rho \left( \frac{\partial u^h}{\partial t} + \nabla \cdot (u^h u^h) - f^h \right) dQ \\
+ \int_{Q_n} & \varepsilon(w^h) : \sigma(u^h, p^h) dQ - \int_{\partial \Omega_n} w^h \cdot h^h dP \\
+ \int_{Q_n} & q^h \nabla \cdot u^h dQ + \int_{\Omega_n} (w^h)^+ \cdot \rho \left( (u^h)^+ - (u^h)^- \right) d\Omega \\
+ \sum_{e=1}^{(n_a)_e} & \int_{Q_n^e} \tau_{\text{SUPS}} \frac{1}{\rho} \left[ \rho \left( \frac{\partial w^h}{\partial t} + u^h \cdot \nabla w^h \right) \\
+ \nabla q^h \right] \cdot r_M(u^h, p^h) dQ \\
+ \sum_{e=1}^{(n_a)_e} & \int_{Q_n^e} \nu_{\text{LSIC}} \nabla \cdot w^h \rho r(C(u^h)) dQ \\
+ \sum_{e=1}^{(n_a)_e} & \int_{Q_n^e} \tau_{\text{SUPS}} r_M(u^h, p^h) \cdot (\nabla w^h) \cdot u^h dQ \\
- & \sum_{e=1}^{(n_a)_e} \int_{Q_n^e} \frac{\tau^2}{\rho} r_M(u^h, p^h) \cdot (\nabla w^h) \cdot r_M(u^h, p^h) dQ \\
= & 0,
\end{align*}
\]

Full discretization of a moving-mesh FSI formulation leads to coupled, nonlinear equation systems that need to be solved at every time step. The FSI coupling technique determines how the coupling between the equation blocks representing the fluid mechanics, structural mechanics and mesh moving equations is handled. It is essential to have a robust coupling method, especially when the structure is light and therefore very sensitive to the changes in the fluid mechanics forces. The coupling method used with the DSD/SST formulation evolved over the years from block-iterative coupling to a more robust version of block-iterative coupling, to quasi-direct coupling and direct coupling (see [5] for the terminology). The quasi-direct and direct coupling methods are applicable to cases with nonmatching fluid and structure meshes at the interface, become equivalent to monolithic methods when the interface meshes are matching, and yield more robust algorithms for FSI computations where the structure is light.

\section{3 Special methods}

A certain class of FSI problems might involve some specific computational challenges beyond those encountered in a typical FSI problem. That requires development of special FSI methods targeting those challenges. A good number of special methods were developed in conjunction with the core ST FSI method to address the specific computational challenges involved in parachute FSI [97],

\[
\int_{Q_n} w^h \cdot \rho \left( \frac{\partial u^h}{\partial t} + \nabla \cdot (u^h u^h) - f^h \right) dQ \\
+ \int_{Q_n} \varepsilon(w^h) : \sigma(u^h, p^h) dQ - \int_{\partial \Omega_n} w^h \cdot h^h dP \\
+ \int_{Q_n} q^h \nabla \cdot u^h dQ + \int_{\Omega_n} (w^h)^+ \cdot \rho \left( (u^h)^+ - (u^h)^- \right) d\Omega \\
+ \sum_{e=1}^{(n_a)_e} \int_{Q_n^e} \tau_{\text{SUPS}} \frac{1}{\rho} \left[ \rho \left( \frac{\partial w^h}{\partial t} + u^h \cdot \nabla w^h \right) \\
+ \nabla q^h \right] \cdot r_M(u^h, p^h) dQ \\
+ \sum_{e=1}^{(n_a)_e} \int_{Q_n^e} \nu_{\text{LSIC}} \nabla \cdot w^h \rho r(C(u^h)) dQ \\
- \sum_{e=1}^{(n_a)_e} \int_{Q_n^e} \tau_{\text{SUPS}} r_M(u^h, p^h) \cdot (\nabla w^h) \cdot u^h dQ \\
- \sum_{e=1}^{(n_a)_e} \int_{Q_n^e} \frac{\tau^2}{\rho} r_M(u^h, p^h) \cdot (\nabla w^h) \cdot r_M(u^h, p^h) dQ \\
= & 0.
\]
patient-specific arterial FSI [92], aerodynamics of flapping wings [51; 52], and wind-turbine aerodynamics [56]. The details on these special methods can be found in the references cited above. Here we give two examples.

3.1 Homogenization model for ringsail parachutes

Parachute FSI involves all the computational challenges of a typical FSI problem. Spacecraft parachutes are most of the time very large ringsail parachutes that are made of a large number of gores, where a gore is the slice of the canopy between two radial reinforcement cables running from the parachute vent to the skirt (see Figure 6). Ringsail parachute gores are constructed from rings and sails, resulting in a parachute canopy with hundreds of ring gaps and sail slits (see Figure 7). The complexity created by this geometric porosity makes FSI modeling inherently challenging.

The Homogenized Modeling of Geometric Porosity (HMGP) [5] and its new version, “HMGP-FG” [97], were introduced to help us bypass the intractable complexities of the geometric porosity by approximating it with an equivalent, locally varying homogenized porosity. In HMGP-FG, the normal velocity crossing the parachute canopy under a pressure differential $\Delta p$ is modeled as

$$u_n = -(k_F)_J \frac{A_F}{A_1} \Delta p - (k_G)_J \frac{A_G}{A_1} \text{sgn}(\Delta p) \sqrt{\frac{\Delta p}{\rho}}, \quad (4)$$

where $A_1$, $A_F$ and $A_G$ are defined in Figure 8, and $(k_F)_J$ and $(k_G)_J$ are the homogenized porosity coefficients for each patch $J$, calculated in a one-time fluid mechanics computation with an $n$-gore slice of the parachute canopy (see Figure 9). Even in a fully open configuration, the parachute canopy goes through a periodic breathing motion where the diameter varies between its minimum and maximum values. The shapes and areas of the gaps and slits vary significantly during this breathing motion (see Figure 10). The porosity coefficients have very good invariance properties with respect to these shape and area changes, and this can be seen in Figure 11.

3.2 Flapping-wing motion representation with higher-order temporal functions

Computer modeling of the aerodynamics of flapping wings requires an accurate temporal representation of the motion and deformation of the wings. It also requires robust and efficient ways of moving the mesh and remeshing...
Fig. 10: The shapes and the areas of the slits vary significantly during the canopy breathing motion.

Fig. 11: The porosity coefficients \((k^F)_J\) and \((k^G)_J\) for each patch \(J\), at different canopy shapes during the breathing motion. The plots show good invariance for these coefficients with respect to the shape changes.

Fig. 12: Mesh motion is represented by using NURBS basis functions in time. The temporal-control meshes are the coefficients of the NURBS basis functions.

Fig. 13: Remeshing is handled by multiple knot insertion where we want to remesh. That point in time becomes a patch boundary.

4 Examples

The first example, a parachute computation, serves the purpose of comparing our computed results to data from drop tests with a base parachute design and gaining confidence in our parachute FSI model. Figure 14 shows the parachute shape and flow field at an instant during the computation and the comparison with the test data. With that confidence, we can do simulation-based design studies [97], such as evaluating the aerodynamic performance of the parachute as a function of the suspension line length (see Figure 15).

Spacecraft parachutes are typically used in clusters of two or three parachutes. The contact between the canopies of the parachute cluster is a computational challenge that we have addressed recently (see [97]). Figure 16 shows a cluster of three parachutes at three different instants during the FSI computation, with contact between two of the parachutes.
Spacecraft parachutes are also typically used in multiple stages, starting with a “reefed” stage where a cable along the parachute skirt constrains the diameter to be less than the diameter in the subsequent stage. After a certain period of time during the descent, the cable is cut and the parachute “disreefs” (i.e. expands) to the next stage. Computing the parachute shape at the reefed stage and FSI modeling during the disreefing involve additional computational challenges created by the increased geometric complexities and by the rapid changes in the parachute geometry. Figure 17 shows such a disreefing (see [99]).

As an additional computational challenge, the ringsail parachute canopy might, by design, have some of its panels and sails removed. The purpose is to increase the aerodynamic performance of the parachute. In FSI computation of parachutes with such “modified geometric porosity,” the flow through the “windows” created by the removal of the panels and the wider gaps created by the removal of the sails cannot be accurately modeled with the HMGP and needs to be actually resolved. This challenge was successfully addressed in the computations reported in [102]. Figure 18 shows a cluster of three parachutes with modified geometric porosity, at an instant during the FSI computation.
Computer modeling of wind-turbine aerodynamics is challenging because correct aerodynamic torque calculation requires correct separation-point calculation, which requires an accurate flow field, which in turn requires good mesh resolution and turbulence model. We describe from [54] how we computed the aerodynamics of an actual wind-turbine rotor by using the DSD/SST-SUPS and DSD/SST-VMST methods. Figure 19 shows, from [23], the airfoil cross-sections of the wind-turbine blade superposed on the blade. Figure 20 shows the full wind-turbine rotor. Figure 21 shows the vorticity magnitude, computed with the DSD/SST-VMST method. In that figure, the blue and yellow correspond to low and high vorticity values, and lighter and darker shades of those two colors correspond to lower and higher values. Figure 22 shows time history of the aerodynamic torque generated by a single blade, as computed with the DST/SST-SUPS, DSD/SST-VMST, and ALE methods. Figure 23 shows aerodynamic torque contribution from the patches defined along the blade.

Including the tower in the model (see Figure 24) increases the computational challenge because of the fast, rotational relative motion between the rotor and tower. We address this additional challenge in [56] by using NURBS basis functions for the temporal representation of the rotor motion, mesh motion and also in remeshing. This is essentially the same computational technology described in Section 3.2 for modeling the aerodynamics of flapping wings. We named this “ST/NURBS Mesh Update Method (STNMUM)” in [56]. Figure 25 shows, from [56], the vorticity magnitude, computed with the DST/SST-VMST method and the STNMUM. In that figure, the color range from blue to red corresponds to a vorticity range from low to high, and lighter and darker shades of a color correspond to lower and higher values.
Fig. 21: Vorticity magnitude, computed with the DSD/SST-VMST method.

Fig. 22: Time history of the aerodynamic torque generated by a single blade. Computed with the DST/SST-SUPS ("SUPS"), DST/SST-VMST ("VMST"), and ALE methods.

Fig. 23: Top: Blade patches. Bottom: Torque contribution from each patch. Computed with the DST/SST-SUPS, DST/SST-VMST, and ALE methods.

Fig. 24: Wind-turbine rotor and tower from [56].

Fig. 25: Vorticity magnitude, computed with the DST/SST-VMST method and the STNMUM (see [56]).
Patient-specific computer modeling of arterial FSI has many challenges. They include calculating an estimated zero-pressure arterial geometry, specifying the velocity profile at an inflow boundary with non-circular shape, using variable arterial wall thickness, building layers of refined fluid mesh near the arterial walls, proper calculation of the wall shear stress (WSS) and oscillatory shear index (OSI), and properly scaling the flow rate at the inflow boundary. Special techniques developed to address these challenges can be found in [92]. Here we present some computations from [92] for cerebral arteries with aneurysm.

Figure 26 shows the arterial lumen geometry obtained from voxel data for three arterial models we consider: Model 1, Model 2, and Model 3. Figure 27 shows, as an example, the fluid mechanics mesh for Model 3, including the layers of refined mesh near the arterial wall. Figure 28 shows, for the three models, the streamlines when the flow rate is maximum. Figures 29 and 30 show the WSS and OSI for Model M6ACom from [92].

As a last set of examples, we present from [51; 52] computational aerodynamics modeling of flapping wings of an actual locust and an MAV. The motion and deformation data for the wings is extracted from the high-speed, multi-camera video recordings of a locust in a wind tunnel at Baylor College of Medicine (BCM), Houston, Texas. The video recording is accomplished by using a set of tracking points marked on the forewings (FW) and hindwings (HW) of the locust. The tracking points can be seen in Figure 31. How the wing motion and deformation data is extracted from the video data and represented using NURBS basis functions in space and time is described in detail in [51]. Figures 32 and 33 show the wind tunnel photographs and the computational model at eight points in time. Figure 34 show how the body and wings compare for the locust and MAV models, and Figure 35 shows the length scales involved in the computations with those models.

Figure 36 shows the streamlines from the locust computation. Figures 37 and 38 show for the locust the vorticity magnitude during the second flapping cycle. Figures 39 and 40 show for the MAV the vorticity magnitude during the third flapping cycle. In Figures 37–40, the color range from blue to red corresponds to a vorticity range from low to high, and lighter and darker shades of a color correspond to lower and higher values. Figures 41–43 show the lift and thrust generated by the locust and MAV.

5 Concluding remarks

Bringing solution and analysis to specific classes of FSI problems with a common computational technology need motivated the development of our core ST FSI technique, its recent versions, and the special ST FSI techniques
targeting specific classes of problems, such as parachute FSI and aerodynamics of flapping wings. We presented an overview of the core and special ST FSI techniques. We also presented examples of different classes of challenging problems solved: spacecraft parachute FSI, wind-turbine aerodynamics, patient-specific arterial FSI, and aerodynamics of flapping wings of an actual locust and an MAV. The examples show that in a diverse set of engineering applications, with the scope and power afforded by the core and special ST FSI techniques, we can provide reliable analysis and support the design process.

**Acknowledgment**

This work was supported in part by NASA JSC grant NNX13AD87G. Method development and evaluation components of the work on aerodynamics of flapping wings and wind-turbine aerodynamics were supported in part by ARO Grant W911NF-12-1-0162 (second author) and Rice–Waseda research agreement (first author). We thank Professor Fabrizio Gabbiani and Dr. Raymond Chan (Baylor College of Medicine) for providing us the digital data extracted from the wind-tunnel videos of the locust.

**References**

2. T.E. Tezduyar, M. Behr, and J. Liou, “A new strategy for finite element computations involving mov-
Fig. 31: Tracking points in the data set from the BCM wind tunnel.

Fig. 32: Comparison of computational model and wind tunnel photographs at first four points in time. Viewing angles are matched approximately. Wind tunnel photographs provided by BCM collaborators.
Fig. 33: Comparison of computational model and wind tunnel photographs at last four points in time. Viewing angles are matched approximately. Wind tunnel photographs provided by BCM collaborators.

Fig. 34: Locust body and wings (left) and MAV body and wings (right).

Fig. 35: Length scales involved in the computations with the locust (top) and MAV (bottom) models.

Fig. 36: Locust. Streamlines colored by velocity magnitude in m/s at approximately 25% (top) and 50% (bottom) of the second flapping cycle.


Fig. 37: Locust. Vorticity magnitude for the first four of eight equally-spaced points during the second flapping cycle.

Fig. 38: Locust. Vorticity magnitude for the last four of the eight equally-spaced points during the second flapping cycle.

Fig. 39: MAV. Vorticity magnitude for the first four of eight equally-spaced points during the third flapping cycle.

Fig. 40: MAV. Vorticity magnitude for the last four of eight equally-spaced points during the third flapping cycle.
Fig. 41: Total lift (top) and thrust (bottom) generated over one cycle.

Fig. 42: Lift (top) and thrust (bottom) generated on the right FW.

Fig. 43: Lift (top) and thrust (bottom) generated on the right HW.


