An Energy Preserving Monolithic Fluid-Structure Finite Element Method

Frédéric Hecht, Olivier Pironneau

An Energy Preserving Monolithic Fluid-Structure Finite Element Method

Frédéric Hecht and Olivier Pironneau

Abstract When written in an Eulerian frame, the conservation laws of continuum mechanic are similar for fluids and solids leading to a single set of variables for a monolithic formulation; the only difference is in the expression of the stress tensors. Such monolithic formulations are well adapted to large displacement fluid-structure configurations, but stability is a challenging problem because of moving geometries. In this article the method and its discretization are presented, stability is discussed for an implicit in time finite element method in space by showing that energy decreases with time. The key numerical ingredient is the Galerkin-Characteric method coupled with a powerful mesh generator. A numerical section discusses implementation issues and presents a few simple tests.

AMS classification

65M60 (74F10 74S30 76D05 76M25).

Introduction

Arbitrary Lagrangian Eulerian methods (ALE) are popular and efficient in the case of small displacements [21]. For large displacements the difficulty is transferred to the mesh [16] and to a lesser extent to the matching conditions at the fluid-solid interface[14]. Furthermore, iterative solvers for FSI which rely on alternative solutions of the fluid and the structure parts are subject to the added mass effect and require special solvers [10][5].

Sorbonne Universités, UPMC (Paris VI) Laboratoire Jacques-Louis Lions e-mail: frederic.hecht@upmc.fr,olivier.pironneau@upmc.fr
Working paper

Immersed boundary methods (IBM)[19] are essentially monolithic and very efficient for thin structures. For solid of zero codimension in a fluid, precision seems to require mesh adaption[8], yet it is a very flexible method shown stable and convergent[3].

Alternative studies to ALE and IBM are few. One old method [2] has resurfaced recently under the name *actualized Lagrangian methods* for computing structures [13][17] (see also [7] although different from the present study because it deals mostly with membranes). Then if such studies are motivated by practical needs there is a future for Eulerian methods.

In an Eulerian approach continuum mechanics doesn't distinguish between solids and fluids till it comes to the constitutive equations. This has been exploited in several studies but most often in the context of ALE[15][23], with one notable exception [9].

In the present study, which is a continuation of [20], we investigate what Stephan Turek [23] calls a monolithic formulation but here in an Eulerian framework, as in [9][22], following the displaced geometry of the fluid and the solid. In [9] the authors experienced meshing difficulties with Lagrangian derivatives which we hope to solve here by using the Characteristic-Galerkin method.

The method is presented in two dimensions for Mooney-Rivlin incompressible materials coupled to incompressible Newtonian flows but it extends to compressible materials and to three dimensions.

1 Continuum Mechanics

Consider two non overlapping time dependent regular bounded open sets of $\mathbb{R}^2$, Ω_t^f and Ω_t^s. Let Ω_t be the interior of $\overline{\Omega}_t^f \cup \overline{\Omega}_t^s$; Ω_t^s will represent the structural domain at time t and Ω_t^f the fluid domain. The fluid-structure interface is denoted $\Sigma_t = \overline{\Omega}_t^f \cap \overline{\Omega}_t^s$ and the boundary of Ω_t is $\partial\Omega_t$. Later we will denote Γ the part of $\partial\Omega_t$ where either the structure is clamped or the fluid does not slip (see figure 1) At initial time Ω_0^f and Ω_0^s are prescribed. The following notations are standard in continuum mechanics [6],[18],[2],[23],[15],[1]:

- $\mathbf{X} : \Omega_0 \times (0,T) \mapsto \mathbf{X}(x_0,t) \in \Omega_t$, the Lagrangian position at t of x_0.
- $\mathbf{u} = \partial_t \mathbf{X}$, the velocity of the deformation,
- $\mathbf{F} = \nabla^t \mathbf{X} = ((\partial_{x_{0_i}} \mathbf{X}_j))$, the transposed gradient of the deformation,
- $J = \det_{\mathbf{F}}$.

We denote by tr_A and $\det_A$ the trace and determinant of A. To describe the system, let

- the density $\rho(x,t) = \mathbf{1}_{\Omega_t^f} \rho^f(x,t) + \mathbf{1}_{\Omega_t^s} \rho^s(x,t)$, at $x \in \Omega_t,\ t \in (0,T)$,

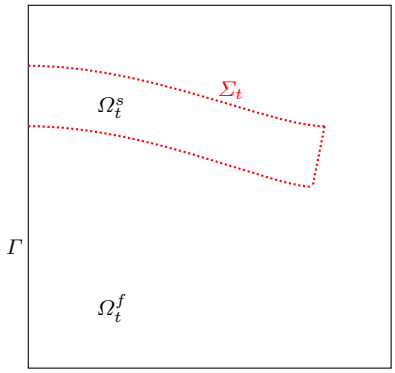

Fig. 1 Sketch of the fluid - Structure domain and notations

- the stress tensor $\sigma(x,t) = \mathbf{1}_{\Omega_t^f}\sigma^f(x,t) + \mathbf{1}_{\Omega_t^s}\sigma^s(x,t)$,
- $\mathbf{d} = \mathbf{X}(x_0,t) - x_0$, the displacement.

Finally and unless specified all spatial derivatives are with respect to $x \in \Omega_t$ and not with respect to $x_0 \in \Omega_0$. If ϕ is a function of $x = \mathbf{X}(x_0,t),\ x_0 \in \Omega_0$,

$$\nabla_{x_0}\phi = [\partial_{x_{0i}}\phi] = [\partial_{x_{0i}}\mathbf{X}_j\partial_{x_j}\phi] = \mathbf{F}^T\nabla\phi.$$

When $\mathbf{X}$ is one-to-one and invertible, $\mathbf{d}$ and $\mathbf{F}$ can be seen implicitly as functions of (x,t) instead of (x_0,t). They are related by

$$\mathbf{F}^T = \nabla_{x_0}\mathbf{X} = \nabla_{x_0}(\mathbf{d}+x_0) = \nabla_{x_0}\mathbf{d}+\mathbf{I} = \mathbf{F}^T\nabla\mathbf{d}+\mathbf{I}, \quad \Rightarrow \quad \mathbf{F} = (\mathbf{I}-\nabla\mathbf{d})^{-T}$$

Time derivatives are related by

$$\mathbb{D}_t\phi := \frac{d}{dt}\phi(\mathbf{X}(x_0,t),t) = \partial_t\phi(x,t) + \mathbf{u}\cdot\nabla\phi(x,t).$$

It is convenient to introduce a notation for the symmetric gradient

$$\nabla^+\mathbf{u} = \nabla\mathbf{u} + \nabla^t\mathbf{u}.$$

Conservation of momentum and conservation of mass take the same form for the fluid and the solid:

$$\rho\mathbb{D}_t\mathbf{u} = \mathbf{f} + \nabla\cdot\sigma, \qquad \frac{d}{dt}(J\rho) = 0,$$

So $J\rho = \rho_0$ at all times and, consequently,

$$J^{-1}\rho_0\mathbb{D}_t\mathbf{u} = \mathbf{f} + \nabla\cdot\sigma \text{ in } \Omega_t,\ \forall t \in (0,T), \tag{1}$$

with continuity of $\mathbf{u}$ and of $\sigma\cdot\mathbf{n}$ at the fluid-structure interface Σ in absence of external surface force. There are also unwritten constraints pertaining to

the realizability of the map $\mathbf{X}$ (see [6],[18]). Finally incompressibility implies $J = 1$ and so $\rho = \rho_0$ along the Lagrangian trajectories. In particular if ρ_0 is piecewise constant and equal to ρ^f, ρ^s in the fluid and the solid, at initial time then it remains constant and equal to ρ^f, ρ^s at later times.

1.1 Constitutive Equations

In this work we shall consider only two dimensional hyperelastic incompressible Mooney-Rivilin materials and Newtonian incompressible fluids.

- For a Newtonian incompressible fluid : $\sigma = -p^f\mathbf{I} + 2\mu^f\nabla^+\mathbf{u}$
- For an hyperelastic incompressible material : $\sigma = -p^s\mathbf{I} + \rho^s\partial_{\mathbf{F}}\Psi\mathbf{F}^T$

where Ψ is the Helmholtz potential which, in the case of a Mooney-Rivlin two dimensional material, is [6]

$$\Psi(\mathbf{F}) = c_1(\mathrm{tr}_{\mathbf{F}^T\mathbf{F}} - 2) + c_2(\mathrm{tr}_{(\mathbf{F}^T\mathbf{F})^2} - \mathrm{tr}^2_{\mathbf{F}^T\mathbf{F}} - 2).$$

1.2 The Mooney-Rivlin 2D Stress Tensor

It is easy to see that

$$\begin{aligned}
&\partial_{\mathbf{F}}\mathrm{tr}_{\mathbf{F}^T\mathbf{F}} = ((\partial_{\mathbf{F}_{ij}}\sum_{m,n} F^2_{m,n})) = 2\mathbf{F}\\
&\partial_{\mathbf{F}}\mathrm{tr}_{(\mathbf{F}^T\mathbf{F})^2} = ((\partial_{\mathbf{F}_{ij}}\sum_{n,m,p,k} F_{n,k}F_{n,m}F_{p,m}F_{p,k})) = 4\mathbf{F}\mathbf{F}^T\mathbf{F}
\end{aligned} \tag{2}$$

Therefore

$$\Psi(\mathbf{F}) = c_1\mathrm{tr}_{\mathbf{F}^T\mathbf{F}} + c_2(\mathrm{tr}_{(\mathbf{F}^T\mathbf{F})^2} - \mathrm{tr}^2_{\mathbf{F}^T\mathbf{F}}) \Rightarrow \partial_{\mathbf{F}}\Psi = 2c_1\mathbf{F} + c_2(4\mathbf{F}\mathbf{F}^T\mathbf{F} - 4\mathrm{tr}_{\mathbf{F}^T\mathbf{F}}\mathbf{F})$$

Let $\mathbf{B} := \mathbf{F}\mathbf{F}^T = \left((\mathbf{I} - \nabla\mathbf{d})(\mathbf{I} - \nabla\mathbf{d})^T\right)^{-1}$, $b :=\mathrm{det}_{\mathbf{B}}$, $c := \mathrm{tr}_{\mathbf{B}} = \mathrm{tr}_{\mathbf{F}^T\mathbf{F}}$. Then

$$\partial_{\mathbf{F}}\Psi\mathbf{F}^T = (2c_1 - 4c_2c)\mathbf{B} + 4c_2\mathbf{B}^2.$$

Now by the Cayley-Hamilton theorem $\mathbf{B}^2 = c\mathbf{B} - b\mathbf{I}$ so

$$\begin{aligned}
&\partial_{\mathbf{F}}\Psi\mathbf{F}^T = 2c_1\mathbf{B} - 4c_2b\mathbf{I} = 2c_1\mathbf{F}\mathbf{F}^T - 4c_2\mathrm{det}_{\mathbf{F}\mathbf{F}^T}\mathbf{I}\\
&\mathbf{B} = c\mathbf{I} - b\mathbf{B}^{-1} = c\mathbf{I} - b(\mathbf{I} - \nabla\mathbf{d} - \nabla^t\mathbf{d} + \nabla\mathbf{d}\nabla^t\mathbf{d}).
\end{aligned} \tag{3}$$

Hence

$$\partial_{\mathbf{F}}\Psi\mathbf{F}^T = (2c_1(c - b) - 4c_2b)\mathbf{I} + 2c_1b(\nabla^+\mathbf{d} - \nabla\mathbf{d}\nabla^t\mathbf{d})$$

$$= 2c_1 \det_{\mathbf{F}\mathbf{F}^T}(\nabla^+\mathbf{d} - \nabla\mathbf{d}\nabla^t\mathbf{d}) + (2c_1 \mathrm{tr}_{\mathbf{F}\mathbf{F}^T} - (2c_1 + 4c_2)\det_{\mathbf{F}\mathbf{F}^T})\mathbf{I}$$

Thus an incompressible 2D Mooney-Rivlin material will have, for some α, α',

$$\partial_{\mathbf{F}}\Psi\mathbf{F}^T = 2c_1(\mathbf{I} - \nabla\mathbf{d})^{-T}(\mathbf{I} - \nabla\mathbf{d})^{-1} + \alpha\mathbf{I} = 2c_1(\nabla^+\mathbf{d} - \nabla\mathbf{d}\nabla^t\mathbf{d}) + \alpha'\mathbf{I}.$$

2 Monolithic Variational Formulation in 2D

For simplicity we shall consider only homogeneous boundary conditions, namely $\Gamma \subset \partial\Omega$ the part of the boundary on which the solid is clamped or the fluid has a no-slip condition and $\partial\Omega_t \backslash \Gamma$ on which there is no constraint. For incompressible Mooney-Rivlin material with Newtonian incompressible fluid the final fluid-structure formulation in two dimensions is:

Find $(\mathbf{u}, p, \mathbf{d}, \Omega_t^f, \Omega_t^s)$ with $\mathbf{u}_{|\Gamma} = 0$ and $\mathbf{d} = 0$, $\mathbf{u}$ given at $t = 0$ and

$$\begin{aligned}\int_{\Omega_t} \Big[\rho\mathbb{D}_t\mathbf{u}\cdot\hat{\mathbf{u}} - p\nabla\cdot\hat{\mathbf{u}} - \hat{p}\nabla\cdot\mathbf{u} + \mathbf{1}_{\Omega_t^f}\frac{\nu}{2}\nabla^+\mathbf{u} : \nabla^+\hat{\mathbf{u}} \\ + \mathbf{1}_{\Omega_t^s} c_1(\nabla^+\mathbf{d} - \nabla\mathbf{d}\nabla^t\mathbf{d}) : \nabla^+\hat{\mathbf{u}}\Big] = \int_{\Omega_t} f\cdot\hat{\mathbf{u}} \\ \mathbb{D}_t\mathbf{d} = \mathbf{u},\end{aligned} \tag{4}$$

for all $(\hat{\mathbf{u}}, \hat{p})$ with $\hat{\mathbf{u}}_{|\Gamma} = 0$, where Ω_t^s and Ω_t^f are defined incrementally by

$$\frac{d\chi}{\mathrm{d}\tau} = \mathbf{u}(\chi(\tau), \tau), \quad \chi(t) \in \Omega_t^r \;\Rightarrow\; \chi(\tau) \in \Omega_\tau^r \;\; \forall\tau \in (0, T), \; r = s, f$$

At time $t = 0$, Ω_0^r are also given, $r = s, f$.

We have used the notation $\mathbf{B} : \mathbf{C} = \mathrm{tr}_{\mathbf{B}^T\mathbf{C}}$.

2.1 Conservation of Energy

Proposition 1.

$$\frac{d}{dt}\int_{\Omega_t}\frac{\rho}{2}|\mathbf{u}|^2 + \frac{\nu}{2}\int_{\Omega_t^f}|\nabla^+\mathbf{u}|^2 + \frac{d}{dt}\int_{\Omega_0^s}\Psi(\mathbf{I} + \nabla_{x_0}\mathbf{d}^T) = \int_{\Omega_t}\mathbf{f}\cdot\mathbf{u}$$

When Ψ is convex, some regularity can be gained from this equality (see [12] for example).

Proof. Choosing $\hat{\mathbf{u}} = \mathbf{u}$, $\hat{p} = -p$ will give the proposition provided

$$2c_1\int_{\Omega_t^s}(\nabla^+\mathbf{d} - \nabla\mathbf{d}\nabla^t\mathbf{d}) : \nabla^+\partial_t\mathbf{d} = \frac{d}{dt}\int_{\Omega_0^s}\Psi(\nabla_{x_0}\mathbf{X}).$$

By construction

$$\int_{\Omega_t^s} c_1(\nabla^+\mathbf{d}-\nabla\mathbf{d}\nabla^t\mathbf{d}) : \nabla^+\hat{\mathbf{u}} = \int_{\Omega_t^s} (\partial_{\mathbf{F}}\Psi(\mathbf{F})\mathbf{F}^T-\alpha\mathbf{I}) : \nabla^+\hat{\mathbf{u}} = \int_{\Omega_0^s} \partial_{\mathbf{F}}\Psi(\mathbf{F}) : \nabla^+{}_{x_0}\hat{\mathbf{u}}$$

Now as $\frac{d}{dt}\Psi(\mathbf{F}) = \partial_{\mathbf{F}}\Psi(\mathbf{F}) : \partial_t\mathbf{F}$ and $\nabla_{x_0}\mathbf{u}(x_0) = \partial_t\nabla_{x_0}\mathbf{d}(x_0) = \partial_t\mathbf{F}^T(x_0)$,

$$\int_{\Omega_t^s} 2c_1(\nabla^+\mathbf{d} - \nabla\mathbf{d}\nabla^t\mathbf{d}) : \nabla\mathbf{u} = \int_{\Omega_0^s} \frac{d}{dt}\Psi(\mathbf{F}) = \frac{d}{dt}\int_{\Omega_0^s} \Psi(\mathbf{I} + \nabla_{x_0}\mathbf{d}^T).$$

The other terms are standard, in particular, with enough regularity on Ω_t^r,

$$\int_{\Omega_t^r} (\partial_t\mathbf{u} + \mathbf{u}\cdot\nabla\mathbf{u})\cdot\mathbf{u} = \frac{d}{dt}\int_{\Omega_t^r} \frac{|\mathbf{u}|^2}{2}, \quad r = f, s$$

3 Discretization in Time

3.1 Discretization of Total Derivatives

Let $\Omega \subset \mathbb{R}^2$, $\mathbf{u} \in \mathbf{H}_0^1(\Omega) = (H_0^1(\Omega))^2$, $t \in (0,T)$ and $x \in \Omega$. Then let $\chi_{\mathbf{u},x}^t(\tau)$ be the solution at time τ of

$$\dot{\chi}(\tau) = \mathbf{u}(\chi(\tau),\tau) \text{ with } \chi(t) = x.$$

If $\mathbf{u}$ is Lipschitz in space and continuous in time the solution exists. The Galerkin-Charasteric method relies on the concept of total derivative:

$$\mathbb{D}_t\mathbf{v}(x,t) := \frac{d}{d\tau}\mathbf{v}(\chi(\tau),\tau)|_{\tau=t} = \partial_t\mathbf{v} + \mathbf{u}\cdot\nabla\mathbf{v}.$$

Given a time step δt, let us approximate

$$\chi_{\mathbf{u}^{n+1},x}^{(n+1)\delta t}(n\delta t) \approx \mathbb{Y}^{n+1}(x) := x - \mathbf{u}^{n+1}(x)\delta t$$

Note that

$$\mathbb{J}_{n+1} := \det{}_{\nabla\mathbb{Y}^{n+1}} = 1 - \nabla\cdot\mathbf{u}^{n+1}\delta t + \det{}_{\nabla\mathbf{u}^{n+1}}\delta t^2.$$

So adding $\mathbb{J}_{n+1}^{-\frac{1}{2}}$ in the formulation below is only an $O(\delta t^2)$ perturbation when the flow is incompressible, and, subject to sufficient regularities we intend to apply the following with $\mathbf{v} = \rho\mathbf{u}$:

$$\frac{\mathbf{v}^{n+1}(x) - \mathbf{v}^n(\mathbb{Y}^{n+1}(x))\mathbb{J}_{n+1}^{-\frac{1}{2}}(x)}{\delta t} = (\partial_t\mathbf{v} + \mathbf{u}\nabla\mathbf{v})|_{x,t^{n+1}} + O(\delta t)$$

3.2 Variational Form of the Time Discrete Problem

At each time step one must find $\mathbf{u}^{n+1} \in \mathbf{H}^1_0(\Omega_{n+1}), p \in L^2(\Omega_{n+1})$, $\Omega^r_{n+1} \subset \mathbb{R}^2$, $r = s, f$, $\Omega_{n+1} = \Omega^f_{n+1} \cup \Omega^s_{n+1}$, such that $\forall \hat{\mathbf{u}} \in \mathbf{H}^1_0(\Omega_{n+1})$, $\forall \hat{p} \in L^2(\Omega_{n+1})$ the 3 following relations hold:

- $$\int_{\Omega_{n+1}} \Big[\frac{\rho^{n+1}\mathbf{u}^{n+1} - (\rho^n \mathbf{u}^n) \circ \mathbb{Y}^{n+1} \mathbb{J}_{n+1}^{-\frac{1}{2}}}{\delta t} \cdot \hat{\mathbf{u}} - p^{n+1} \nabla \cdot \hat{\mathbf{u}} - \hat{p} \nabla \cdot \mathbf{u}^{n+1} + \mathbf{1}_{\Omega^f_{n+1}} \frac{\nu}{2} \nabla^+ \mathbf{u}^{n+1} : \nabla^+ \hat{\mathbf{u}} + c_1 \mathbf{1}_{\Omega^s_{n+1}} J^{-1}_{n+1} [(\mathbf{I} - \nabla \mathbf{d}^{n+1})^{-T} (\mathbf{I} - \nabla \mathbf{d}^{n+1})^{-1}] : \nabla^+ \hat{\mathbf{u}} \Big] = \int_{\Omega_{n+1}} \mathbf{f} \cdot \hat{\mathbf{u}},$$
- $\Omega_{n+1} = (\mathbb{Y}^{n+1})^{-1}(\Omega_n) = \{x : \ \mathbb{Y}^{n+1}(x) := x - \mathbf{u}^{n+1}(x)\delta t \in \Omega_n\}$
- $\mathbf{d}^{n+1} = \mathbf{d}^n \circ \mathbb{Y}^{n+1} + \delta t \mathbf{u}^{n+1}, \quad J^{-1}_{n+1} = \det_{\mathbf{I} - \nabla \mathbf{d}^{n+1}}$ (5)

Proposition 2. *Problem (5) is equivalent to*

$$\int_{\Omega_{n+1}} \Big[\frac{\rho^{n+1}\mathbf{u}^{n+1} - (\rho^n \mathbf{u}^n) \circ \mathbb{Y}^{n+1} \mathbb{J}_{n+1}^{-\frac{1}{2}}}{\delta t} \cdot \hat{\mathbf{u}} - p^{n+1} \nabla \cdot \hat{\mathbf{u}} - \hat{p} \nabla \cdot \mathbf{u}^{n+1} + \mathbf{1}_{\Omega^f_{n+1}} \frac{\nu}{2} \nabla^+ \mathbf{u}^{n+1} : \nabla^+ \hat{\mathbf{u}} + c_1 \mathbf{1}_{\Omega^s_{n+1}} J^{-1}_{n+1} \mathbb{J}^{-2}_{n+1} [(\mathbf{I} + \nabla \mathbf{u}^{n+1} \delta t)^T (\mathbf{I} - \nabla \tilde{\mathbf{d}}^n)^{-T} (\mathbf{I} - \nabla \tilde{\mathbf{d}}^n)^{-1} (\mathbf{I} + \nabla \mathbf{u}^{n+1} \delta t)] : \nabla^+ \hat{\mathbf{u}} \Big] = \int_{\Omega_{n+1}} \mathbf{f} \cdot \hat{\mathbf{u}}, \quad \Omega_{n+1} = \{x : \ \mathbb{Y}^{n+1}(x) := x - \mathbf{u}^{n+1}(x)\delta t \in \Omega_n\} \tag{6}$$

with $\mathbf{d}$ *updated by* $\mathbf{d}^{n+1} = \tilde{\mathbf{d}}^n + \delta t \mathbf{u}^{n+1}$ *where* $\tilde{\mathbf{d}}^n = \mathbf{d}^n \circ \mathbb{Y}^{n+1}$.

Proof From the definition of $\mathbf{d}^{n+1}$ we have

$$\nabla \mathbf{d}^{n+1} = \nabla \mathbb{Y}^{n+1} \nabla \mathbf{d}^n \circ \mathbb{Y}^{n+1} + \nabla \mathbf{u}^{n+1} \delta t = (\mathbf{I} - \delta t \nabla \mathbf{u}^{n+1}) \nabla \mathbf{d}^n \circ \mathbb{Y}^{n+1} + \nabla \mathbf{u}^{n+1} \delta t$$

Hence

$$\mathbf{I} - \nabla \mathbf{d}^{n+1} = (\mathbf{I} - \nabla \mathbf{u}^{n+1} \delta t)(\mathbf{I} - \nabla \tilde{\mathbf{d}}^n)$$

The identity (in 2D only)

$$(\mathbf{I} - \nabla \mathbf{u}^{n+1} \delta t)^{-1} = \mathbb{J}^{-1}_{n+1} (\mathbf{I} + \nabla \mathbf{u}^{n+1} \delta t)$$

completes the proof ◇

Remark 1. Still (6) must be solve iteratively. One possibility is to change $\mathbf{u}^{n+1}$ in $\{\mathbb{Y}^{n+1}, \Omega_{n+1}, \mathbb{J}_{n+1}, J_{n+1}\}$ into $\tilde{\mathbf{u}}^{n+1}$ and loop, starting from $\tilde{\mathbf{u}}^{n+1} = \mathbf{u}^n$, compute the solution of (6), then update $\tilde{\mathbf{u}}^{n+1}$ to the new value etc, till convergence.

3.3 Stability of the Scheme Discretized in Time

Proposition 3. *The mapping* $\mathbf{X}^n : \Omega_0 \mapsto \Omega_n$ *is given by* $\mathbf{X}^{n+1} = (\mathbb{Y}^{n+1})^{-1} \circ \mathbf{X}^n$, $n \geq 1$ *and the jacobian of the transformation is* $\mathbf{F}^n := \nabla^t_{x_0}\mathbf{X}^n = (\mathbf{I} - \nabla\mathbf{d}^n)^{-T}$.

Proof

Notice that $\mathbb{Y}^1(\mathbb{Y}^2(..\mathbb{Y}^{n-1}(\mathbb{Y}^n(\Omega_n))..)) = \Omega_0$ Hence

$$\mathbf{X}^{n+1} = [\mathbb{Y}^1(\mathbb{Y}^2(..\mathbb{Y}^n(\mathbb{Y}^{n+1})))]^{-1} = (\mathbb{Y}^{n+1})^{-1} \circ \mathbf{X}^n.$$

By definition of $\mathbf{d}^{n+1}$ in (5)

$$\begin{aligned} \mathbf{d}^{n+1}(\mathbf{X}^{n+1}(x_0)) &= \mathbf{d}^n(\mathbb{Y}^{n+1}(\mathbf{X}^{n+1}(x_0))) + \mathbf{u}^{n+1}(\mathbf{X}^{n+1}(x_0))\delta t \\ &= \mathbf{d}^n(\mathbf{X}^n(x_0)) + \mathbf{u}^{n+1}(\mathbf{X}^{n+1}(x_0))\delta t, \end{aligned} \tag{7}$$

so $\mathbf{X}^{n+1}(x_0) = \mathbf{d}^{n+1}(\mathbf{X}^{n+1}(x_0)) + x_0$ and therefore

$$\begin{aligned} \mathbf{F}^{n+1} &= \nabla^t_{x_0}(\mathbf{d}^{n+1}((\mathbf{X}^{n+1}(x_0))) + x_0), \\ &= \nabla\mathbf{d}^{n+1^T}\mathbf{F}^{n+1} + \mathbf{I} \;\Rightarrow\; \mathbf{F}^{n+1} = (\mathbf{I} - \nabla\mathbf{d}^{n+1})^{-T} \end{aligned} \tag{8}$$

◇

Note that (7) shows too:

$$\mathbf{F}^{n+1} = \mathbf{F}^n + \delta t \nabla^T_{x_0}\mathbf{u}^{n+1} \tag{9}$$

Lemma 1.

$$\int_{\Omega^s_{n+1}} c_1[J^{-1}_{n+1}[(\mathbf{I} - \nabla\mathbf{d}^{n+1})^{-T}(\mathbf{I} - \nabla\mathbf{d}^{n+1})^{-1}] : \nabla^+\hat{\mathbf{u}} = \int_{\Omega^s_0} \partial_{\mathbf{F}}\Psi^{n+1} : \nabla_{x_0}\hat{\mathbf{u}}$$

Proof

From Proposition 3,

$$\begin{aligned} &\int_{\Omega^s_{n+1}} c_1[J^{-1}_{n+1}[(\mathbf{I} - \nabla\mathbf{d}^{n+1})^{-T}(\mathbf{I} - \nabla\mathbf{d}^{n+1})^{-1}] : \nabla^+\hat{\mathbf{u}} \\ &\quad= \int_{\Omega^s_{n+1}} c_1 J^{-1}_{n+1}[\mathbf{F}^{n+1}\mathbf{F}^{n+1^T}] : \nabla^+\hat{\mathbf{u}} \\ &\quad= \int_{\Omega^s_0} c_1\mathbf{F}^{n+1} : \nabla^+{}_{x_0}\hat{\mathbf{u}} = \frac{1}{2}\int_{\Omega^s_0} \partial_{\mathbf{F}}\Psi^{n+1} : \nabla^+{}_{x_0}\hat{\mathbf{u}} \end{aligned} \tag{10}$$

◇

Theorem 1. *When* $f = 0$ *and* ρ *is constant in each domain* Ω^r_n, $r = s, f$,

$$\int_{\Omega_n} \frac{\rho^n}{2}|\mathbf{u}^n|^2 + \delta t \sum_{k=1}^{n} \int_{\Omega_k^f} \frac{\nu}{2}|\nabla^+ \mathbf{u}^k|^2 + \int_{\Omega_0^s} \Psi^n \leq \int_{\Omega_0} \frac{\rho^0}{2}|\mathbf{u}^0|^2 + \int_{\Omega_0^s} \Psi^0 (11)$$

Proof Let $r = s$ or f. Let us choose $\hat{\mathbf{u}} = \mathbf{u}^{n+1}$ in (5). By Schwartz inequality

$$\int_{\Omega_{n+1}^r} \sqrt{\mathbb{J}}_{n+1} \mathbf{u}^n \circ \mathbb{Y}^{n+1} \cdot \mathbf{u}^{n+1} \leq \left(\int_{\Omega_{n+1}^r} \mathbb{J}_{n+1} (\mathbf{u}^n \circ \mathbb{Y}^{n+1})^2 \right)^{\frac{1}{2}} \left(\int_{\Omega_{n+1}^r} {\mathbf{u}^{n+1}}^2 \right)^{\frac{1}{2}}$$

By a change of variable

$$\int_{\Omega_{n+1}^r} \mathbb{J}_{n+1} (\mathbf{u}^n \circ \mathbb{Y}^{n+1})^2 = \int_{\Omega_n} {\mathbf{u}^n}^2$$

Consequently, using $ab \leq \frac{1}{2}a^2 + \frac{1}{2}b^2$,

$$\int_{\Omega_{n+1}^r} \sqrt{\mathbb{J}}_{n+1} \mathbf{u}^n \circ \mathbb{Y}^{n+1} \cdot \mathbf{u}^{n+1} \leq \frac{1}{2} \int_{\Omega_n^r} {\mathbf{u}^n}^2 + \frac{1}{2} \int_{\Omega_n^r} {\mathbf{u}^{n+1}}^2$$

Finally,

$$\int_{\Omega_{n+1}} \frac{\rho^{n+1}}{2}|\mathbf{u}^{n+1}|^2 + \delta t \int_{\Omega_{n+1}^f} \frac{\nu}{2}|\nabla^+ \mathbf{u}^{n+1}|^2 + \int_{\Omega_0} \Psi^{n+1} \leq \int_{\Omega_n} \frac{\rho^n}{2}|\mathbf{u}^n|^2 + \int_{\Omega_0} \Psi^n (12)$$

3.4 Spatial Discretization with Finite Elements

Let $\mathcal{T}_h^0$ be a triangulation of the initial domain. Spatial discretization can be done with Lagrangian triangular elements of degree 2 for the space V_h of velocities and displacements and Lagrangian triangular elements of degree 1 for the pressure space Q_h provided that the pressure be different in the structure and the fluid because the pressure is discontinuous at the interface Σ; therefore Q_h is the space of piecewise linear functions on the triangulation continuous in $\Omega_{n+1}^r,\ r = s, f$. A small penalization with parameter ϵ must be added to impose uniqueness of the pressure.

Discretization in space by the Finite Element Method leads to find $\mathbf{u}_h^{n+1}, p_h^{n+1} \in V_{0h} \times Q_h$ such that for all $\hat{\mathbf{u}}_h, \hat{p}_h \in V_{0h} \times Q_h$,

$$\int_{\Omega_{n+1}} \Big[\frac{\rho^{n+1}\mathbf{u}_h^{n+1} - (\rho^n \mathbf{u}_h^n) \circ \mathbb{Y}^{n+1} \mathbb{J}_{n+1}^{-\frac{1}{2}}}{\delta t} \cdot \hat{\mathbf{u}}_h - p_h^{n+1} \nabla \cdot \hat{\mathbf{u}}_h - \hat{p}_h \nabla \cdot \mathbf{u}_h^{n+1}$$

$$+\mathbf{1}_{\Omega^f_{n+1}}\frac{\nu}{2}\nabla^+\mathbf{u}^{n+1}:\nabla^+\hat{\mathbf{u}}+2c_1\mathbf{1}_{\Omega^s_{n+1}}J^{-1}_{n+1}\mathbb{J}^{-2}_{n+1}$$
$$[(\mathbf{I}+\nabla\mathbf{u}^{n+1}_h\delta t)^T(\mathbf{I}-\nabla\tilde{\mathbf{d}}^n_h)^{-T}(\mathbf{I}-\nabla\tilde{\mathbf{d}}^n_h)^{-1}(\mathbf{I}+\nabla\mathbf{u}^{n+1}_h\delta t)]:\nabla^+\hat{\mathbf{u}}\Big]$$
$$=\int_{\Omega_{n+1}}\mathbf{f}\cdot\hat{\mathbf{u}}_h,,\quad \Omega_{n+1}=(\mathbb{Y}^{n+1})^{-1}(\Omega_n)=\{x:\ \mathbb{Y}^{n+1}(x)\in\Omega_n\}\qquad(13)$$

with $\mathbf{d}_h$ updated by $\mathbf{d}^{n+1}_h=\tilde{\mathbf{d}}^n_h+\delta t\mathbf{u}^{n+1}_h$where $\tilde{\mathbf{d}}^n_h=\mathbf{d}^n_h\circ\mathbb{Y}^{n+1}$. and where

$$\mathbb{Y}^{n+1}(x)=x-\mathbf{u}^{n+1}_h(x)\delta t$$

The proof for the spatially continuous case will work for the discrete case if

$$\mathbf{X}^n=\mathbf{X}^{n+1}\circ\mathbb{Y}^{n+1}.\qquad(14)$$

First let us consider the P^2-P^1 element for $\in V_{0h}\times Q_h$. Then the mapping $x\mapsto\mathbb{Y}^{n+1}(x)$ is quadratic and it is possible to assume that it preserves the triangles of an isoparametric mesh, meaning that the vertices and the mid-edge nodes of the mesh at time t^{n+1} should be mapped by $\mathbb{Y}^{n+1}$ to the corresponding vertices and mid-edge nodes of the mesh at time t^n. Unfortunately such a construction does not satisfy (14) for the obvious reason that $f\circ g$ is of degree 4 when f and g are quadratic. However for isoparametric elements all computations are done on the reference element so it is quite possible that such a construction works, but he haven't analyzed it.

Now consider the $P^1_3-P^1$ element; the fluid pressure and the solid pressure are continuous and piecewise linear on the triangulation and each "pressure-triangle" is divided into 3 smaller triangles by a a fourth vertex anywhere inside the triangle; on this refined triangulation the velocity is continuous piecewise linear.

Then (14) holds and the proof of the spatially continuous case can be adapted. The inner vertex used to construct the fluid mesh will be moved by $\mathbb{Y}^{n+1}$ but $\mathbf{X}^{n+1}\circ\mathbb{Y}^{n+1}$ remains linear and for each triangle $T^k_n=\mathbb{Y}^{n+1}(T^k_{n+1})$.

4 Numerical Tests

In all our tests we define c_1 in terms of the Young modulus E and the Poisson ratio ω by $c_1=E/(2(1+\omega))$.

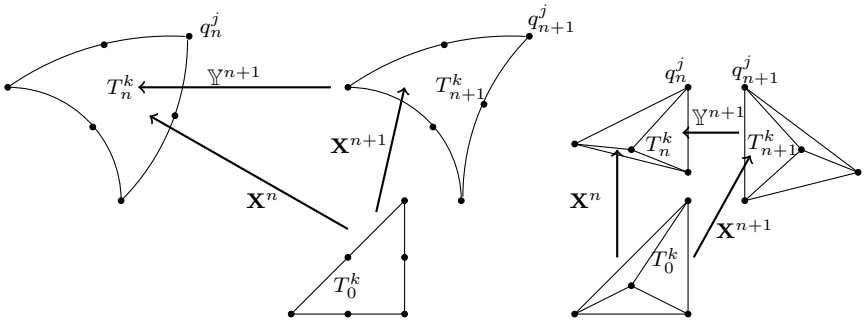

Fig. 2 Sketch to understand if $\mathbf{X}^n = \mathbb{Y}^{n+1}\mathbf{X}^{n+1}$; on the left the case of P^2-isoparametric element for the velocities and on the right the case of the $P_3^1 - P1$ element where each triangle is divided into four subtriangles on which the velocities are P^1 and continuous. A triangle T_0^k in the reference domain (chosen here to be its initial position at time zero) becomes the isoparametric triangles T_n^k and T_{n+1}^k: $T_n^k = \mathbf{X}^n(T_0^k)$ and $T_{n+1}^k = \mathbf{X}^{n+1}(T_0^k)$ in the case of quadratic velocities. But as the vertices and mid-edges of T_n^k are obtained by moving those of T_{n+1}^k by $-\mathbf{u}^{n+1}\delta t$ we cannot have $\mathbb{Y}^{n+1}(T_{n+1}^k) = T_n^k$ because the composition of two quadratic maps is of degree 4 in general. However in the case of affine velocities triangles are transformed into triangles and the mapping composition property holds.

4.1 Validation with a Rotating Disk

The geometry is a disk of radius R=3 with an hyperelastic incompressible solid with $E = 2.15, \omega = 0.29$, giving $c_1 = 0.833$ in a disk of same center and radius $r_s = 1.5$. The outer part is filled by a Newtonian fluid with $\nu = 0.1$. The velocity of the fluid is a rotation on the outer cylinder of magnitude 3. The time varies from 0 to T=10. There are 80 time steps.

Naturally the fluid rotates first and then induce the solide to rotate as well.

As everything is axisymmetric the computation can be done with the reduced problem

$$\rho\partial_t v - \frac{1}{r}\partial_r[\xi r\partial_r v] + \xi\frac{v}{r^2} = 0$$

with $\rho = \mathbf{1}_{|x|\leq r_s}\rho^s + \mathbf{1}_{|x|>r_s}\rho^f$, $\xi = \mathbf{1}_{|x|<=r_s}2c_1 + \mathbf{1}_{|x|>r_s}\nu)$, and with $v_{|x|=R} = 3$. Comparison is given on figure 10.

4.2 Clamped Beam Falling Freely in a Fluid

The fall of an hyperelastic incompressible beam of size 9×1 is studied in a rectangular box of size 10×7 filled with a fluid. As the beam is clamped on the right and free to fall under its own weight in so doing it compels

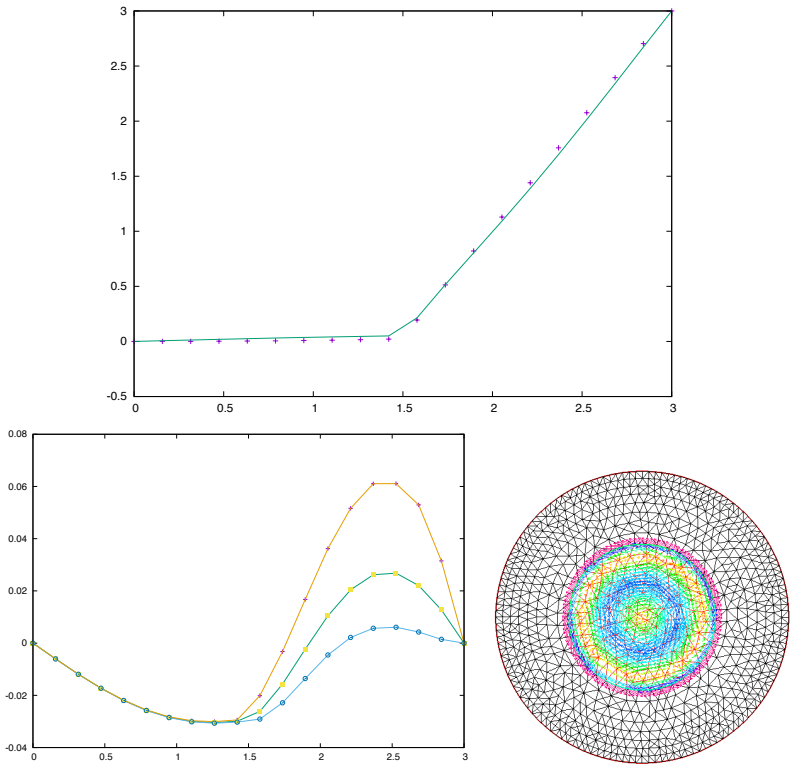

Fig. 3 *Rotating cylinder filled partially with a fluid and partially with an hyperelastic solid. After a while the solid rotates (yet slowly), carried by the rotation of the fluid. Top: comparison at $T = 5$ of the vertical velocity on the horizontal axis for $x \in (0, R)$, after 100 time steps for the coarsest triangulation compared with the axisymmetric 1D solution. Bottom: for 3 triangulations (m=1,2,3 in the freefem script) on the left, errors versus x: on the right the velocity in the solid is shown. The radius of the solid part is $R/2$.*

the fluid to move; the fluid is also subject to gravity. The results are shown on figure 4. The Mooney-Rivlin coefficient is computed with $E = 5.3$ and $\omega = 0.29$. Its density is 1 and the gravity force is -0.5. The fluid has $\rho^f = 0.5$ and $\nu = 0.1$. The computation stops at $T = 200$ after 100 time steps with $\delta t = 2$. The vertices of the mesh in the solid part are moved by their own velocity, i.e. $\mathbf{u}^{n+1}\delta t$ by calling `movemesh()`, the mesh moving function of `FreeFem++`. At each time iteration the mesh is updated twice; first by the new velocities and then readjusted by the velocities recomputed on the new mesh. A procedure to extract the modified solid boundary has been implemented and from this knowledge the mesh in the fluid region is rebuilt every time

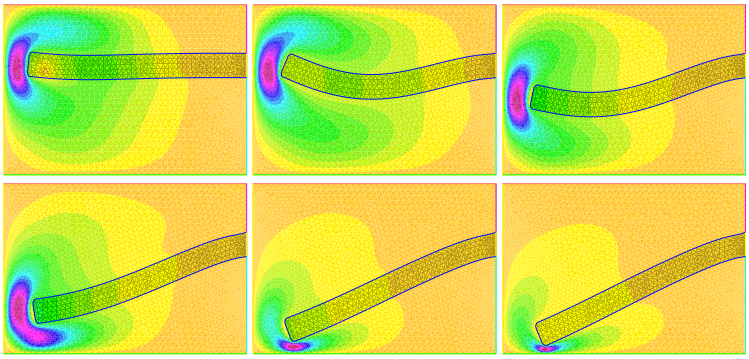

Fig. 4 *Free fall of the same beam, clamped on the right, in a fluid initially at rest. The norm of velocities and the mesh are shown at* $t = 1, 20, 40, 60, 80, 100$.

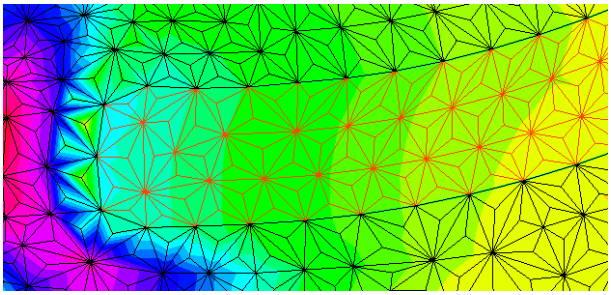

Fig. 5 *Free fall of a beam, clamped on the right, in a fluid initially at rest. The 3P1-P1 element with non centered inner vertex is shown in action. Zoom on the left tip of the beam showing that not all inner vertices are in the center. The colors correspond to the norm of the velocity vectors.*

step by the `FreeFem++` module `buildmesh()` which is based on a Delaunay-Voronoi algorithm.

Three elements have been tested: P^2-P^1, $3P^1-P^1$ with fixed mid vertices and $3P^1 - P^1$ with moving mid vertices.

The second example is the free motion of an hyperelastic incompressible solid submitted to the force due to the rotation of a fluid around it, in a disk, induced by the sliding of the lower horizontal boundary at unit speed. Initially the solid is a disk. The fluid domain is a rectangle of size 10×7 and $\nu = 0.1$, $\rho^f = 0.5$ while $\rho^s = 1$. The gravity is -0.2. Here the mesh in the fluid region is moved by solving $\Delta \mathbf{u} = 0$ with appropriate boundary

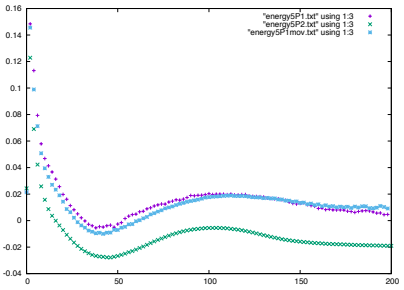

Fig. 6 *Free fall of a beam, clamped on the right, in a fluid initially at rest. Energy versus time. The relative difference between the energy of the system (kinetic + fluid viscous dissipation + potential energy of the structure) and the initial energy plus the work of the gravity .*

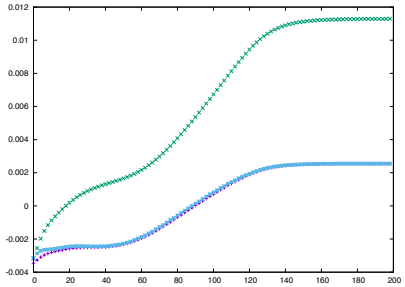

Fig. 7 *Free fall of a beam, clamped on the right, in a fluid initially at rest. Relative errors versus time of a) the area of the beam (purple cross), J (green cross), $\mathbb{J}$ (blue star). Computations are done with the P2-P1 element. Notice that the errors are quite small.*

conditions and remeshed occasionally when the result is too distorted. The method is not so robust as the previous one.

4.3 Beam Attached Behind a Cylinder

This test is similar to the one proposed in [23] except that the flagella is incompressible so it cannot be compared with the results in [23]. The characteristics are: Cylinder radius $r = 0.05$, height of pipe $H = 0.41$,length of pipe $L = 2.5$, length and thickness of flagella $l = 0.175$, $h_2 = 0.01$, kinematic viscosity $\nu = 0.002$, parabolic inflow velocity with $U = 0.4$ at the center, $\rho^s = \rho^f = 1$, $E = 12.9$ (i.e. $c_1 = 5$). Time step $\delta t = 0.02$. Number of vertices 1500.

Figure 10 shows the pressure maps and the position of the beam at $t = 1, 1.8, 2.36$ and $t = 3$, while figure 9 shows the position of the right tip of the beam as a function of time. The FreeFem++ script is given in Appendix C.

Conclusion

A fully Eulerian fluid-structure formulation has been presented and an attempt at deriving an implicit unconditionally stable monolithic finite element discretization has been proposed and studied. The method has been implemented with `FreeFem++`. It is reasonably robust; occasionally it stalls in the

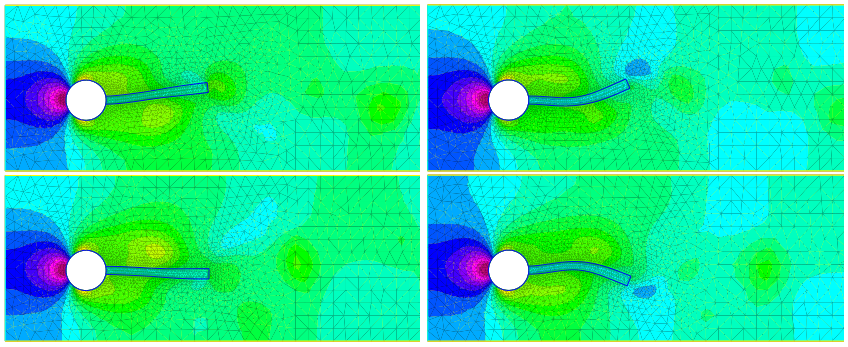

Fig. 8 *Pressure map in a zoomed region for an hyperelastic flagella attached to a sphere in a Navier-Stokes flow. After a while the Von Karman alley of vertices destabilizes the flagella which begins to beat. Note that the incompressibility coefficient used implies a rather soft material and so the first effect of the flow is to elongate the beam. The snapshots correspond to* $t = 5.4, 5.6, 5.8$ *and* $t = 6.0$

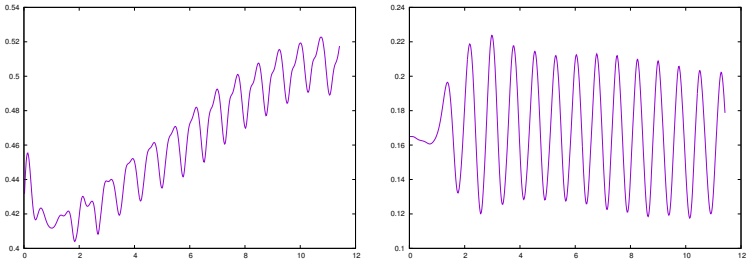

Fig. 9 *On the left and on the right* x *and* y *position of the lower corner point of the flagella versus time. Notice how the flagella stretches on account of the pull received from the flow*

module `movemesh()` in the solid part when an element is turned over by the moving velocity. In principle such situation could be fixed by call ing a remeshing module; however the numerical tests showed us than when this happens the trouble is deeper and a few time iteration later the same trouble will arise, indicating in fact that the flaw is perhaps in the realizability of the experiment. Consequently more analysis is needed to find what are the conditions for stability.

References

1. S.S. Antman. Nonlinear Problems of Elasticity (2^{nd} ed.). Springer series in Applied Mathematical Sciences, Vol. 107, 2005.
2. K.J. Bathe. Finite element procedures, Prentice-Hall, Englewood Cliffs, New-Jersey. 1996.
3. D. Boffi, N. Cavallini, L. Gastaldi, The finite element immersed boundary method with distributed Lagrange multiplier, arXiv:1407.5184v2, 2015 (to appear)
4. K. Boukir, Y. Maday, B. Metivet. A high order characteristics method for the incompressible Navier-Stokesequations. Comp. Methods in Applied Mathematics and Engineering 116 (1994), 211-218.
5. M. Bukaca, S. Canic, R. Glowinski, J. Tambacac, A. Quainia. Fluidstructure interaction in blood flow capturing non-zero longitudinal structure displacement. Journal of Computational Physics 235 (2013) 515541.
6. P.G. Ciarlet. Mathematical Elasticity. North Holland, 1988.
7. G.H. Cottet, E. Maitre, T. Milcent. Eulerian formulation and level set models for incompressible fluid-structure interaction. M2AN Math. Model. Numer. Anal. 42 (2008), no. 3, 471492.
8. Th. Coupez, L. Silva, E. Hachem, Implicit Boundary and Adaptive Anisotropic Meshes, S. Peretto, L. Formaggia eds. *New challenges in Grid Generation and Adaptivity for Scientific Computing*. SEMA-SIMAI Springer series, vol 5, 2015.
9. Th. Dunne, Adaptive Finite Element Approximation Of Fluid-Structure Interaction Based On An Eulerian Variational Formulation, ECCOMAS CFD 2006 P. Wesseling, E. Oñate and J. Périaux (Eds) Elsevier, TU Delft, The Netherlands, 2006.
10. M. A. Fernandez, J. Mullaert, M. Vidrascu. Explicit Robin-Neumann schemes for the coupling of incompressible fluids with thin-walled structures, Comp. Methods in Applied Mech. and Engg. 267, 566593, 2013.
11. F. Hecht New development in FreeFem++, J. Numer. Math., 20 (2012), pp. 251-265. www.FreeFem.org.
12. P. Hauret. Méthodes numériques pour la dynamique des structures non-linéaires incompressibles à deux échelles. Doctoral thesis, Ecole Polytechnique, 2004.
13. S. Léger. Méthode lagrangienne actualisée pour des problèmes hyperélastiques en très grandes déformations. Thèse de doctorat, Université Laval, 2014.
14. P. Le Tallec and P. Hauret. Energy conservation in fluid-structure interactions. In P. Neittanmaki Y. Kuznetsov and O. Pironneau, editors, Numerical methods for scientific computing, variational problems and applications, CIMNE, Barcelona, 2003.
15. P. Le Tallec and J. Mouro. Fluid structure interaction with large structural displacements. Comp. Meth. Appl. Mech. Eng., 190(24-25) :3039-3068, 2001.
16. J. Liu. A second-order changing-connectivity ALE scheme and its application to FSI with large convection of fluids and near-contact of structures. Submitted to Journal of Computational Physics, (2015).
17. I-Shih Liu, R. Cipolatti and M. A. Rincon. Incremental linear approximation for finite elasticity. Proc ICNAAM 2006, Wiley.
18. J. Marsden and T.J.R. Hughes Mathematical Foundations of Elasticity. Dover publications 1993.
19. C.S. Peskin. The immersed boundary method. Acta Numerica, 11:479-517, 2002.
20. O. Pironneau. *Numerical Study of a Monolithic Fluid-Structure Formulation.* i *Variational Analysis and Aerospace Engineering: Mathematical Challenges for the Aerospace of the Future* Aldo Frediani, Bijan Mohammadi, Olivier Pironneau (eds). Springer 2016.
21. L. Formaggia, A. Quarteroni, A. Veneziani. *Cardiovasuclar Mathematics.* Springer MS&A Series. Springer-Verlag, 2009.
22. R. Rannacher and T. Richter, An Adaptive Finite Element Method for Fluid-Structure Interaction Problems Based on a Fully Eulerian Formulation. Springer Lecture Notes in Computational Science and Engineering, Vol 73, 2010.

23. J. Hron and S. Turek. A monolithic fem solver for an ALE formulation of fluidstructure interaction with configuration for numerical benchmarking. European Conference on Computational Fluid Dynamics ECCOMAS CFD 2006 P. Wesseling, E. Onate and J. Periaux (Eds). TU Delft, The Netherlands (2006)

Appendix A: the FreeFem++ script that gave figure 4

```
// FSI with same variable for fluid and structure
// move the sold mesh and remesh the fluid
// iterate once (or not if interdom=0) on the position of the solid.
load "Curvature"
load "isoline"
verbosity=0;
int la1=10;
border a(t=10,3)  { x=0; y=t;};  //  left
border b(t=0,10) { x=t; y=3 ;};  //  bottom
border c(t=3,7)  { x=10; y=t ;}; //  right low
border d(t=19,0) {if(t<=9){ x=10-t; y=7;}  //  low beam
else if(t<=10){ x=1; y=t-2;  }  // left beam
  else { x=t-9; y=8; }  // top beam
  label=la1; }
border g(t=8,10) { x=10; y=t;};  // right up
border f(t=10,0) { x=t; y=10 ;}; //  top
border e(t=0,1) { x=10; y=7+t;};  // right beam
int m=1;

func FixBord=a(m*30)+b(m*20)+c(m*16)+g(m*5)+f(m*20);  //plot( FixBord+d(m*50)+e(4*m),wait=1);
mesh th = buildmesh( FixBord+d(m*50)+e(4*m));
int nsl=1;
real lga1,lga2;
real[int,int] SLa1(3,nsl);
lga1=extractborder(th,la1,SLa1);

border a11(t=0,SLa1.m-1){    P.x=SLa1(0,t) ;P.y=SLa1(1,t) ;   label=la1;}
th=buildmesh( a11((SLa1.m-1))+FixBord+e(4*m),fixeborder=1);
int[int] rr=[0,2];
th=change(th,region=rr);
rr=[1,0];th=change(th,region=rr);
int fluid=th(1,1).region, beam=th(9,7.5).region;
//cout<<fluid<<"  "<<beam<<endl;
mesh ths=trunc(th,region==beam);//plot(ths,wait=1);
mesh thf=trunc(th,region==fluid);//plot(thf,wait=1);
mesh thsold=ths;
fespace V2h(th,P2);
fespace V2hsold(thsold,P2);
fespace Vh(th,P1);
fespace V2hs(ths,P2);

Vh p,ph,pp,pph;
V2h u,v,uh,vh, uold=0, vold=0;
V2hs d1=0,d2=0, dd1,dd2, usold=0,vsold=0,us,vs; // used to keep data on an old mesh
V2hsold do1,do2, uuold,vvold,uu,vv;

real nu=0.1;
real E = 2.15*2;
real sigma = 0.29;
real mu = E/(2*(1+sigma));
real c1=mu;
real penal=1e-6;
//real lambda = E*sigma/((1+sigma)*(1-2*sigma));
real gravity = -0.5;
real rhof=0.5, rhos=1.;

macro div(u,v) ( dx(u)+dy(v) ) // EOM
macro DD(u,v)  [[2*dx(u),div(v,u)],[div(v,u),2*dy(v)]] // EOM
macro Grad(u,v)[[dx(u),dy(u)],[dx(v),dy(v)]] // EOM
macro det(u,v) (dx(u)*dy(v)-dx(v)*dy(u)) //EOM
macro ONE(a) [[1,0],[0,1]] //EOM
int NN=100;
real T=200, dt=T/NN;

problem aa([u,v,p,pp],[uh,vh,ph,pph]) =
int2d(th,beam)( rhos*[u,v]'*[uh,vh]/dt - div(uh,vh)*pp  - div(u,v)*pph+ penal*pp*pph + penal*p*ph
+
(1-div(d1,d2)+det(d1,d2))*
dt*c1*trace(DD(uh,vh)*(DD(u,v)  -Grad(u,v)*Grad(d1,d2)' - Grad(d1,d2)*Grad(u,v)')))
   + int2d(th,beam) ( -rhos*gravity*vh +
   (1-div(d1,d2)+det(d1,d2))*
c1*trace(DD(uh,vh)*(DD(d1,d2) - Grad(d1,d2)*Grad(d1,d2)'))
   -  rhos*[usold,vsold]'*[uh,vh]/dt)
+  int2d(th,fluid)(rhof*[u,v]'*[uh,vh]/dt- div(uh,vh)*p -div(u,v)*ph + penal*p*ph +  penal*pp*pph
+  nu/2*trace(DD(uh,vh)'*DD(u,v)))
   -  int2d(th,fluid)(rhof*gravity*vh +rhof*[convect([uold,vold],-dt,uold),convect([uold,vold],-dt,vold)]'*[uh,vh]/dt)
+ on(a,b,c,e,f,g,u=0,v=0) ;

// Computation time loop
real workR=0*int2d(th,beam)(2*c1), workL=workR;
bool iterdom=1;
for(int n=0;n<=NN;n++){
   dd1=d1;dd2=d2;
  if(iterdom){
      thsold=ths;
```

```
      do1=d1; do2=d2;uuold=usold;vvold=vsold;
      aa;
      uu=u; vv=v;
     ths = movemesh(thsold,[x+uu*dt,y+vv*dt]); // update th by moving thold
     lga1=extractborder(ths,la1,SLa1);
   thf=buildmesh( a11(-(SLa1.m-1))+FixBord,fixeborder=1);
   th=ths+thf;
   usold=0;usold[]=uuold[];vsold=0;vsold[]=vvold[];
   d1=0;d1[]=do1[]; d2=0; d2[]=do2[];
   uold=uold; vold=vold;
     }
      aa;
      us=u;vs=v;
    ths = movemesh(thsold,[x+us*dt,y+vs*dt]); // update th by moving thold
         d1=0;  d2=0;  usold=0;   vsold=0;
         // copy old values into d1,d2 defined with new th (does d o X^n)
         d1[]=dd1[]+dt*us[]; // update array computed on old mesh
         d2[]=dd2[]+dt*vs[];
      usold[]=us[];
      vsold[]=vs[];   // does u o X^n
  lga1=extractborder(ths,la1,SLa1);
  thf=buildmesh( a11(-(SLa1.m-1))+FixBord,fixeborder=1);
  th=ths+thf;
  u=u; v=v; uold=u;vold=v;
if((n==1)||(n/20)*20==n){
vh= sqrt(u^2+v^2); plot(th,vh,fill=1,value=0, ps="vitesse"+n+".eps");}
   workR += int2d(th,beam)(gravity*v*dt) +  int2d(th,fluid)(rhof*gravity*v*dt);
   workL += int2d(th,fluid)(dt*nu*(dx(u)^2+dy(u)^2+dx(v)^2+dy(v)^2));
 real energy=workL+int2d(th,fluid)(rhof*(u*u+v*v)/2)
 +int2d(th,beam)(rhos*(u*u+v*v)/2+c1*trace(DD(dd1,dd2)*(DD(d1,d2) - Grad(d1,d2)*Grad(d1,d2)')));
  if(n==0) cout<<" time, area, energy,  work+energy0"<<endl<<n*dt<<" "<<int2d(th,beam)(1.)
  <<" "<<energy<<" "<< workR<<endl; else cout<<n*dt<<" "<<int2d(th,beam)(1.)<<" "<<
  1-energy/(workR+1e-6)<<" "<< workR<<endl;
// cout<<n*dt<<" "<<int2d(th,beam)(1./9)-1<<"  "
//<<int2d(th,beam)((1-div(d1,d2)+det(d1,d2))/9)-1<<" "<< int2d(th,beam)((1-div(u,v)*dt+det(u,v)*dt^2)/9)-1<<endl;
}
```

5 Appendix B: The FreeFem script for the rotating disk

```
// FSI with same variable for fluid and structure
// hyperelastic solid in a rotating fluid due to outer boundary rotation
load "Curvature"
load "isoline"
verbosity=0;
real R=1.5, RR=3;
int m=3,
n, I=20, NN=80;
real dr = RR/(I-1),eps=1e-15, T=5, dt=T/NN;

mesh Th=square(100,5,[RR*x,0.1*y]);
fespace Wh(Th,P2,periodic=[[1,x],[3,x]]);
fespace W0(Th,P1dc);

real nu=0.1;
real E = 2.15;
real sigma = 0.29;
real mu = E/(2*(1+sigma));
real c1=mu,  penal=1e-5;
//real lambda = E*sigma/((1+sigma)*(1-2*sigma));
real  rhof=1., rhos=2;

Wh d=0,w,wh,wold=0;
W0 nnu=nu*(x>R)+2*c1*dt*(x<=R), Rho=rhof*(x>R)+rhos*(x<=R);
////////////////////////////////////////////////////////////

problem AA(w,wh,init=n) = int2d(Th)(Rho*(x+eps)*w*wh/dt+(x+eps)*nnu*dx(w)*dx(wh) + nnu*w*wh/(x+eps))
- int2d(Th)(Rho*(x+eps)*wold*wh/dt - 2*c1*(x<R)*((x+eps)*dx(d)*dx(wh) + d*wh/(x+eps)))
+on(2,w=RR);
for(n=0;n<=NN;n++){ AA; wold=w; d=d+w*dt;}
// plot(w,dim=3,fill=true,value=true,wait=1);

int la1=10,la2=11;
// solid region
border a1(t=0, pi)  { x=R*cos(t); y=R*sin(t) ;label=la1;};        //  left
border a2(t=pi, 2*pi)  { x=R*cos(t); y=R*sin(t) ;label=la2;};     //  right
// Fluid region
border b(t=0,2*pi) { x=RR*cos(t); y=RR*sin(t) ;label=1;};

func FixBord=b(m*50);
mesh th = buildmesh( a1(m*30)+a2(m*30)+FixBord);
```

```
int nsl=1;
real lga1,lga2;
real[int,int] SLa1(3,nsl), SLa2(3,nsl);
lga1=extractborder(th,la1,SLa1);
lga2=extractborder(th,la2,SLa2);

border a11(t=0,SLa1.m-1){    P.x=SLa1(0,t) ;P.y=SLa1(1,t) ;   label=la1;}
border a22(t=0,SLa2.m-1){    P.x=SLa2(0,t) ;P.y=SLa2(1,t) ;   label=la2;}
th=buildmesh( a11((SLa1.m-1))+a22((SLa1.m-1))+FixBord,fixeborder=1);

int fluid=th(2*R,0).region, beam=th(0,0).region;
mesh ths=trunc(th,region==beam);
mesh thsold=ths;
mesh thf=trunc(th,region==fluid);

fespace V2h(th,P2);
fespace Vh(th,P1);
fespace V2hs(ths,P2);
fespace V2hsold(thsold,P2);

Vh p,ph,pp,pph;
V2h u,v,uh,vh, uold=0, vold=0;
V2hs d1=0,d2=0, usold,vsold;
V2hsold do1,do2, uso,vso,uo,vo;// used to keep data on an old mesh



macro div(u,v) ( dx(u)+dy(v) ) // EOM
macro DD(u,v)  [[2*dx(u),div(v,u)],[div(v,u),2*dy(v)]] // EOM
macro Grad(u,v)[[dx(u),dy(u)],[dx(v),dy(v)]] // EOM

problem aa([u,v,p,pp],[uh,vh,ph,pph]) =
int2d(th,beam)(  rhos*[u,v]'*[uh,vh]/dt - div(uh,vh)*p  - div(u,v)*ph
+ penal*p*ph + penal*pp*pph
   + dt*c1*trace(DD(uh,vh)*(DD(u,v)  -Grad(u,v)*Grad(d1,d2)'
   - Grad(d1,d2)*Grad(u,v)'))
  + int2d(th,beam)(  c1*trace(DD(uh,vh)*(DD(d1,d2) - Grad(d1,d2)*Grad(d1,d2)'))
   - rhos*[usold,vsold]'*[uh,vh]/dt)
  + int2d(th,fluid)( rhof*[u,v]'*[uh,vh]/dt- div(uh,vh)*pp -div(u,v)*pph
       + penal*p*ph + penal*pp*pph + nu/2*trace(DD(uh,vh)'*DD(u,v)))
  - int2d(th,fluid)( rhof*[convect([uold,vold],-dt,uold),
   convect([uold,vold],-dt,vold)]'*[uh,vh]/dt )
  + on(1,v=x,u=-y) ;

// Computation time loop
bool iterdom=1;  // 1 for an implicit computation of the solid domain
ofstream myfile("/Users/pironneau/Desktop/v"+m+".txt");
ofstream myfile2("/Users/pironneau/Desktop/v"+m+"g.txt");
for(n=1;n<=NN;n++){
  if((n/300)*300==n){
   th=adaptmesh(th,u,v,err=0.0001);//0.3/m,IsMetric=1);
   ths=trunc(th,region==beam);
thf=trunc(th,region==fluid); thsold=ths;
   //plot(th); plot(ths);plot(thf,wait=1);
   uold=uold; vold=vold;
   d1=d1;d2=d2; usold=usold;vsold=vsold;
   }
   thsold=ths;
   do1=d1; do2=d2;
   uso=usold;vso=vsold;
  if(iterdom){
      aa;
      uo=u; vo=v;
     ths = movemesh(thsold,[x+uo(x+uo(x,y)*dt/2,y+vo(x,y)*dt/2)*dt,
     y+vo(x+uo(x,y)*dt/2,y+vo(x,y)*dt/2)*dt]);
     lga1=extractborder(ths,la1,SLa1);
lga2=extractborder(ths,la2,SLa2);
   thf=buildmesh( a11(-(SLa1.m-1))+a22(-(SLa2.m-1))+FixBord,fixeborder=1);
   th=ths+thf;
   usold=0;usold[]=uso[];
   vsold=0;vsold[]=vso[];
   d1=0; d1[]=do1[];
   d2=0; d2[]=do2[];
   uold=uold; vold=vold;
     }
     aa;
      uo=u; vo=v;
     ths = movemesh(thsold,[ x+uo(x+uo(x,y)*dt/2,y+vo(x,y)*dt/2)*dt,
     y+vo(x+uo(x,y)*dt/2,y+vo(x,y)*dt/2)*dt]);
     lga1=extractborder(ths,la1,SLa1);
lga2=extractborder(ths,la2,SLa2);
   thf=buildmesh( a11(-(SLa1.m-1))+a22(-(SLa2.m-1))+FixBord,fixeborder=1);
   th=ths+thf;
   usold=0; usold[]=uso[];
   vsold=0; vsold[]=vso[];
   real xm=int2d(ths)(x), ym=int2d(ths)(y), um=int2d(ths)(uo), vm=int2d(ths)(vo),
   nm=int2d(ths)(x^2+y^2);
   real yum=int2d(ths)(y*uo), xvm=int2d(ths)(x*vo), onem=int2d(ths)(1.);
```

```
   real omega=-(yum-xvm-ym*um/onem+xm*vm/onem) / (nm-(ym^2+xm^2)/onem);
   uo=uo+omega*y-(um+omega*ym)/onem;
   vo=vo-omega*x-(vm-omega*xm)/onem;
   d1=0; d1[]=do1[]+uo[]*dt;
   d2=0; d2[]=do2[]+vo[]*dt;
    uold=u;vold=v;
// if((n/10)*10==n)
vh= sqrt(u^2+v^2); plot(th,[u,v],fill=0,value=1);
cout<<n*dt<<" = time,  area= "<<int2d(th,beam)(1.)<<endl;
for(int i=0;i<I;i++) myfile<< v(i*dr,0) <<"  ";
myfile<<endl;
}
for(int i=0;i<I;i++) myfile2<< i*dr<<"  "<< v(i*dr,0)<<"  "
    <<w(i*dr,0.05)<<"  "<<v(i*dr,0)-w(i*dr,0.05) <<endl;
```

Appendix C: A flagella behind a circle

The freefem script is given below.

```
// Turek FSI Test
load "Curvature"
load "isoline"
verbosity=0;
int la1=10, la2=11;
real cx0 = 0.2, cy0 = 0.2-0.025; // center of cyl.
real xa = 0.15, ya=0.2, xe = 0.25,ye =0.2;// point for pressure..
real r=0.05, H=0.41, L=2.5;
real ll=0.35/2, h2=0.01;// =h/2
real la=asin(h2/r), x0=sqrt(r*r-h2*h2);

border fr1(t=0,L){x=t; y=0; label=1;}
border fr2(t=0,H){x=L; y=t; label=2;}
border fr3(t=L,0){x=t; y=H; label=1;}
border fr4(t=H,0){x=0; y=t; label=3;}
border fr5(t=la,2*pi-la){x=cx0+r*cos(-t); y=cy0+r*sin(-t); label=4;}
border br1(t=-la,la){x=cx0+r*cos(-t); y=cy0+r*sin(-t); label=4;}
border br2(t=0,ll){x=cx0+x0+t; y=cy0-h2;label=la1;}
border br3(t=-h2,h2){x=x0+cx0+ll; y=cy0+t;label=la1;}
border br4(t=ll,0){x=cx0+x0+t; y=cy0+h2;label=la1;}
int m=3;
func FixBord = fr1(20*m)+fr2(4*m)+fr3(20*m)+fr4(4*m)+fr5(12*m);
//plot(FixBord + br1(m)+br2(6*m)+br3(m)+br4(6*m));
mesh th=buildmesh(FixBord + br1(m)+br2(12*m)+br3(m)+br4(12*m));// plot(th,wait=1);

real rhos=1, rhof=1, nu=1./5000, c1=0.5, Ubar=0.2*2, penal=1e-6;

int nsl=1;
real lga1;
real[int,int] SLa1(3,nsl);
lga1=extractborder(th,la1,SLa1);
border a11(t=0,SLa1.m-1){ P.x=SLa1(0,t); P.y=SLa1(1,t); label=la1;}
th=buildmesh( a11((SLa1.m-1))+br1(m)+FixBord,fixeborder=1); //plot(th,wait=1);
int[int] rr=[0,2]; th=change(th,region=rr);
int fluid=th(0.01,0.01).region, beam=th(x0+cx0+0.1,cy0).region;
cout<<fluid<<"  "<<beam<<endl;
mesh ths=trunc(th,region==beam);// plot(ths,wait=1);
mesh thf=trunc(th,region==fluid);// plot(thf,wait=1);
mesh thsold=ths;

fespace V2h(th,P2);
fespace V2hsold(thsold,P2);
fespace Vh(th,P1);
fespace V2hs(ths,P2);

Vh p,ph,pp,pph;
V2h u,v,uh,vh, uold=0, vold=0;
V2hs d1=0,d2=0, dd1,dd2, usold=0,vsold=0; // used to keep data on an old mesh
V2hsold do1,do2, uusold,vvsold,us,vs;

macro div(u,v) ( dx(u)+dy(v) ) // EOM
macro DD(u,v)  [[2*dx(u),div(v,u)],[div(v,u),2*dy(v)]] // EOM
macro Grad(u,v)[[dx(u),dy(u)],[dx(v),dy(v)]] // EOM

int NN=5000;
real T=100, dt=T/NN;

problem aa([u,v,p,pp],[uh,vh,ph,pph]) =
int2d(th,beam)( rhos*[u,v]'*[uh,vh]/dt - div(uh,vh)*pp  - div(u,v)*pph+ penal*pp*pph+ penal*p*ph
+dt*c1*trace(DD(uh,vh)*(DD(u,v)  -Grad(u,v)*Grad(d1,d2)' - Grad(d1,d2)*Grad(u,v)')))
   + int2d(th,beam) ( c1*trace(DD(uh,vh)*(DD(d1,d2) - Grad(d1,d2)*Grad(d1,d2)'))
   -  rhos*[usold,vsold]'*[uh,vh]/dt)
```

```
+  int2d(th,fluid)(rhof*[u,v]'*[uh,vh]/dt- div(uh,vh)*p -div(u,v)*ph + penal*p*ph + penal*pp*pph
+ nu/2*trace(DD(uh,vh)'*DD(u,v)))
   -  int2d(th,fluid)(rhof*[convect([uold,vold],-dt,uold),convect([uold,vold],-dt,vold)]'*[uh,vh]/dt)
+ on(1,4, u=0,v=0) +  on(3,u=Ubar*y*(0.41-y)*6/0.1681,v=0);

// Computation time loop
bool iterdom=0;
for(int n=1;n<NN;n++){
/* if((n/400)*400==n){
th=adaptmesh(th,0.2/m,IsMetric=1);
ths=trunc(th,region==beam); thsold=ths;
    thf=trunc(th,region==fluid);
//plot(th); plot(ths);plot(thf,wait=1);
uold=uold; vold=vold;
d1=d1;d2=d2; usold=usold;vsold=vsold;
}
*/ thsold=ths;
dd1=d1;dd2=d2;
if(iterdom){
do1=d1; do2=d2;uusold=usold;vvsold=vsold;
aa;
us=u; vs=v;
  ths = movemesh(thsold,[x+us*dt,y+vs*dt]); // update th by moving tholdlga1=extractborder(ths,la1,SLa1);
  lga1=extractborder(ths,la1,SLa1);
  thf=buildmesh( a11(-(SLa1.m-1))+FixBord,fixeborder=1);
  th=ths+thf; // plot(th,wait=1);
  usold=0;usold[]=uusold[];vsold=0;vsold[]=vvsold[];
  d1=0;d1[]=do1[]; d2=0; d2[]=do2[];
  uold=uold; vold=vold;
}
aa;
us=u;vs=v;
ths=movemesh(thsold,[x+us*dt,y+vs*dt]); // update th by moving thold
d1=0;  d2=0;  usold=0;   vsold=0;// copy old values into d1,d2 defined with new th (does d o X^n)
d1[]=dd1[]+dt*us[]; // update array computed on old mesh
d2[]=dd2[]+dt*vs[];
usold[]=us[];
vsold[]=vs[];   // does u o X^n
lga1=extractborder(ths,la1,SLa1);
// plot( a11((-SLa1.m-1))+FixBord,wait=1); plot(ths,wait=1);
thf=buildmesh( a11(-(SLa1.m-1))+FixBord,fixeborder=1);//  plot(thf,wait=1);
th=ths+thf; //   plot(th,wait=1);
// ths=trunc(th,region==beam); plot(ths,wait=1);
th=change(th,region=rr);
//     thf=trunc(th,region==fluid); plot(thf,wait=1);

u=u; v=v; uold=u;vold=v;
// if((n/10)*10==n)
vh= sqrt(u^2+v^2);
func box=[[-0.1,-0.1],[L/3,H+0.1]];
plot(th,p,bb=box,fill=1,value=0,wait=0,WindowIndex=0);
plot(th,vh,fill=1,value=0,wait=0,WindowIndex=1);
cout<<n*dt<<" = time,  area= "<<int2d(th,beam)(1.)<<endl;
}
```

Appendix D: A valve

The geometry is a rectangular 2D beam in a square duct. The prescribed difference of stress from upper to lower part (almost equal to the pressure difference) is sinusoidal. It steers an oscillating movement in the beam which closes and opens the canal to the flow. Some snapshot of the flow are given below. The freefem scripts are given below.

```
// FSI for a valve}
load "Curvature"
load "isoline"
verbosity=0;
int la1=10;
real L=-3, R0=7, R1=8;
border a(t=10,L)  { x=0; y=t;};  //  left
border b(t=0,10) { x=t; y=L ;};  //  bottom
border c(t=L,7)  { x=10; y=t ;}; //  right low
border d(t=19,0) {if(t<=9){ x=10-R0*sin(t*pi/18); y= 7-R0+R0*cos(t*pi/18);}//x=10-t; y=7;} //  low beam
else if(t<=10){ x=10-R1+(R1-R0)*(10-t)/2; y=7-R0;  }  // left beam
  else { x=10-R1*sin((19-t)*pi/18); y= 8-R1+R1*cos((19-t)*pi/18);}//x=t-9; y=8; } // top beam
label=la1; }
```

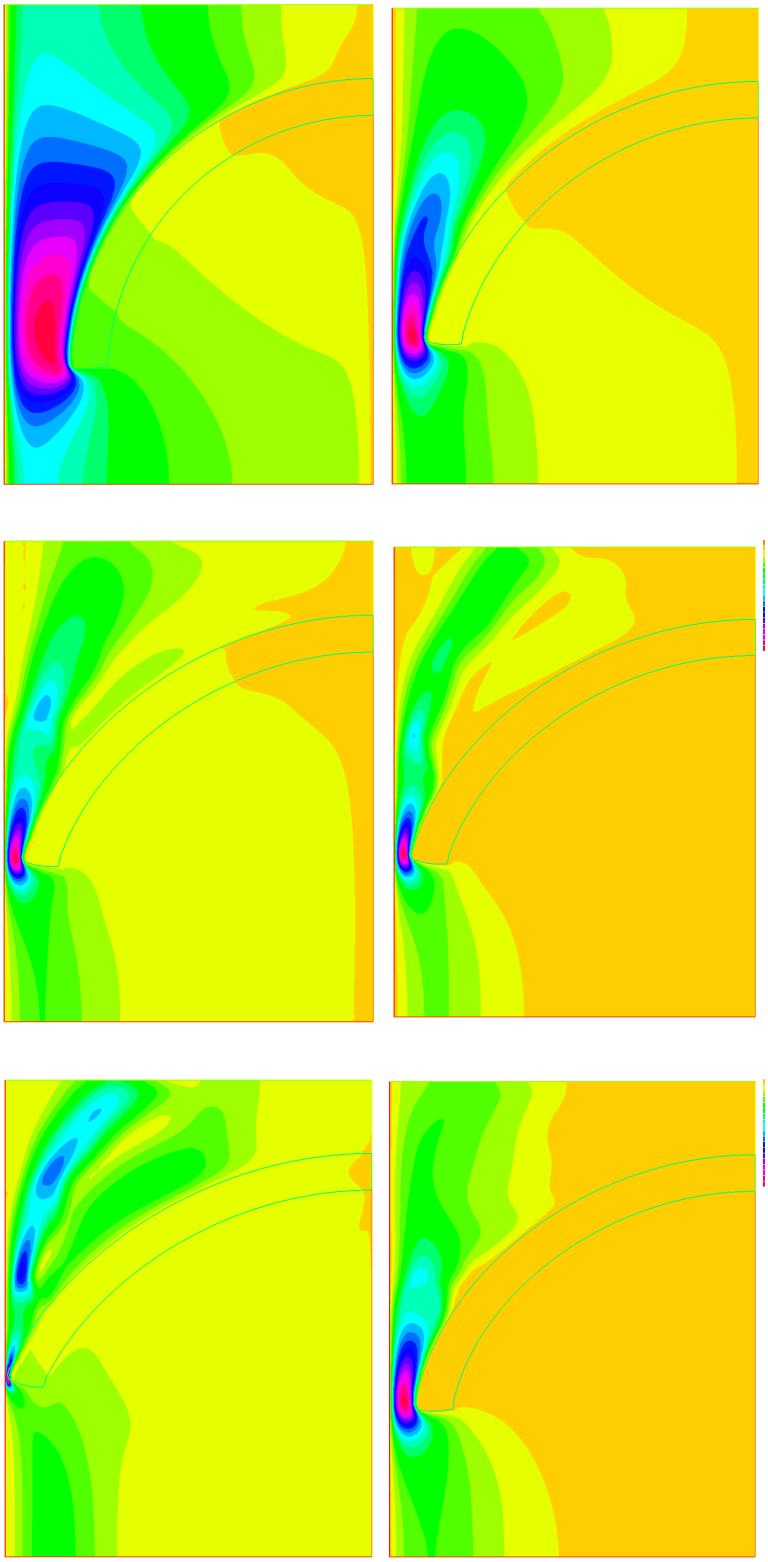

Fig. 10 *Hyperelastic quarter-circular beam in an oscillating flow, causing its solid part to close the pipe and stop the flow but then when the flow reverse the beam moves and reopens the pipe to the flow. Snapshots are for* $t = 20, 100, 180, 240, 300, 4003$

```
border g(t=8,10) { x=10; y=t;};  // right up
border f(t=10,0) { x=t; y=10 ;}; //  top
border e(t=0,1) { x=10; y=7+t;};  // right beam
int m=2;

func FixBord=a(m*30)+b(m*20)+c(m*16)+g(m*5)+f(m*20); // plot( FixBord+d(m*50)+e(4*m),wait=1);
mesh th = buildmesh( FixBord+d(m*50)+e(4*m));  //plot(th,wait=1);
//th=adaptmesh(th,0.25,IsMetric=1) ;
int nsl=1;
real lga1,lga2;
real[int,int] SLa1(3,nsl);
lga1=extractborder(th,la1,SLa1);

border a11(t=0,SLa1.m-1){    P.x=SLa1(0,t) ;P.y=SLa1(1,t) ;   label=la1;}
th=buildmesh( a11((SLa1.m-1))+FixBord+e(4*m),fixeborder=1);
int[int] rr=[0,2];
th=change(th,region=rr);
rr=[1,0];th=change(th,region=rr);
int fluid=th(1,1).region, beam=th(9,7.5).region;
cout<<fluid<<"  "<<beam<<endl;
mesh ths=trunc(th,region==beam);//plot(ths,wait=1);
mesh thf=trunc(th,region==fluid);//plot(thf,wait=1);
mesh thsold=ths;
fespace V2h(th,P2);
fespace V2hsold(thsold,P2);
fespace Vh(th,P1);
fespace V2hs(ths,P2);

Vh p,ph,pp,pph;
V2h u,v,uh,vh, uold=0, vold=0;
V2hs d1=0,d2=0, dd1,dd2, usold=0,vsold=0,us,vs; // used to keep data on an old mesh
V2hsold do1,do2, uuold,vvold,uu,vv;
real nu=0.001;
real E = 2.15;
real sigma = 0.29;
real mu = E/(2*(1+sigma));

func c1=x*x*mu;

real penal=1e-6;
//real lambda = E*sigma/((1+sigma)*(1-2*sigma));
real gravity = -0.;
real rhof=0.5, rhos=1.;

macro div(u,v) ( dx(u)+dy(v) ) // EOM
macro DD(u,v)  [[2*dx(u),div(v,u)],[div(v,u),2*dy(v)]] // EOM
macro Grad(u,v)[[dx(u),dy(u)],[dx(v),dy(v)]] // EOM

int NN=200;
real t,T=4000, dt=T/NN;

problem aa([u,v,p,pp],[uh,vh,ph,pph]) =
int2d(th,beam)( rhos*[u,v]'*[uh,vh]/dt - div(uh,vh)*pp  - div(u,v)*pph+ penal*pp*pph + penal*p*ph
+dt*c1*trace(DD(uh,vh)*(DD(u,v)  -Grad(u,v)*Grad(d1,d2)' - Grad(d1,d2)*Grad(u,v)')))
   + int2d(th,beam) ( -rhos*gravity*vh +c1*trace(DD(uh,vh)*(DD(d1,d2) - Grad(d1,d2)*Grad(d1,d2)'))
   -  rhos*[usold,vsold]'*[uh,vh]/dt)
+  int2d(th,fluid)(rhof*[u,v]'*[uh,vh]/dt- div(uh,vh)*p -div(u,v)*ph + penal*p*ph +  penal*pp*pph
+  nu/2*trace(DD(uh,vh)'*DD(u,v)))
   -  int2d(th,fluid)(rhof*gravity*vh +rhof*[convect([uold,vold],-dt,uold),convect([uold,vold],-dt,vold)]'*[uh,vh]/dt)
   -int1d(th,f)(0.03*sin(20*t/T)*vh)
+ on(a,c,e,g,u=0,v=0) + on(b,u=0)+ on(f,u=0);

// Computation time loop
bool iterdom=1;
for(int n=0;n<NN;n++){  t=n*dt;
   dd1=d1;dd2=d2;
  if(iterdom){
      thsold=ths;
      do1=d1; do2=d2;uuold=usold;vvold=vsold;
      aa;
      uu=u; vv=v;
     ths = movemesh(thsold,[x+uu*dt,y+vv*dt]); // update th by moving thold
     lga1=extractborder(ths,la1,SLa1);
   thf=buildmesh( a11(-(SLa1.m-1))+FixBord,fixeborder=1);
   th=ths+thf;
   usold=0;usold[]=uuold[];vsold=0;vsold[]=vvold[];
   d1=0;d1[]=do1[]; d2=0; d2[]=do2[];
   uold=uold; vold=vold;
     }
      aa;
      us=u;vs=v;
    ths = movemesh(thsold,[x+us*dt,y+vs*dt]); // update th by moving thold
         d1=0;  d2=0;  usold=0;   vsold=0;// copy old values into d1,d2 defined with new th (does d o X^n)
         d1[]=dd1[]+dt*us[]; // update array computed on old mesh
         d2[]=dd2[]+dt*vs[];
       usold[]=us[];
       vsold[]=vs[];   // does u o X^n
  lga1=extractborder(ths,la1,SLa1);
  thf=buildmesh( a11(-(SLa1.m-1))+FixBord,fixeborder=1);
```

```
  th=ths+thf;
  u=u; v=v; uold=u;vold=v;
// if((n/10)*10==n)
vh= sqrt(u^2+v^2); plot(th,vh,fill=1,value=1);
plot(ths,[us,vs],fill=0,value=1, WindowIndex=1);
cout<<n*dt<<" = time,  area= "<<int2d(th,beam)(1.)<<endl;
}
```