Construction of a High Order Fluid-Structure Interaction Solver

Gonçalo Pena[a,b], Christophe Prud'homme[c]

[a] *CMUC, Department of Mathematics, University of Coimbra*
3001 - 454 Coimbra, Portugal
[b] *CMCS, École Polytechnique Fédérale de Lausanne*
MA C2 573 (Bâtiment MA), Station 8, CH-1015 Lausanne, Switzerland
[c] *Laboratoire Jean Kuntzmann, Université Joseph Fourier Grenoble 1*
BP 53 38041 Grenoble Cedex 9, France

Abstract

Accuracy is critical if we are to trust simulation predictions. In settings such as fluid-structure interaction it is all the more important to obtain reliable results to understand, for example, the impact of pathologies on blood flows in the cardiovascular system. In this paper, we propose a computational strategy for simulating fluid structure interaction using high order methods in space and time.

First, we present the mathematical and computational core framework, Life, underlying our multi-physics solvers. Life is a versatile library allowing for 1D, 2D and 3D partial differential solves using h/p type Galerkin methods. Then, we briefly describe the handling of high order geometry and the structure solver. Next we outline the high-order space-time approximation of the incompressible Navier-Stokes equations and comment on the algebraic system and the preconditioning strategy. Finally, we present the high-order Arbitrary Lagrangian Eulerian (ALE) framework in which we solve the fluid-structure interaction problem as well as some initial results.

Key words: h/p Galerkin method, incompressible Navier-Stokes, Preconditioning, Arbitrary Lagrangian Eulerian, Fluid Structure Interaction

1. Mathematical and computational framework

We present a very brief overview of Life, a unified framework for finite element and spectral element methods in 1D, 2D and 3D in C++. The objectives of this framework are to construct a versatile, albeit small and manageable, mathematical kernel in C++ *(i)* easily solving partial differential equations(PDEs) problems, thanks to a syntax close the mathematical abstractions and language, *(ii)* allowing to test and compare different numerical methods, e.g. continuous Galerkin (cG) versus discontinuous Galerkin (dG).

1.1. Basic principles

The syntax, the semantics and the pragmatics of the library are very close to the mathematics and in particular Galerkin type methods. C++ has been the chosen language because it supports very well multiple paradigms design and offers a wide range of solutions

for a given problem. Generic programming, OO programming, meta-programming are such paradigms and they are definitely very useful when dealing with mathematical abstractions. The following C++ code uses all these paradigms

```
integrate( boundaryfaces(mesh), im, gradv(u)*N() ).evaluate();
```

It integrates the scalar function $\nabla u \cdot \boldsymbol{n}$, where $\boldsymbol{n}$ is the outward normal to the boundary of the domain. `im` provides the numerical integration method and `boundaryfaces(mesh)` returns a pair of iterators over the set of faces on the boundary of the domain. We refer the reader to [1, 2] for a more complete overview of Life and a description of the language.

1.2. The polynomial library and embedded language

The polynomial library and the embedded language are the two cornerstones of Life. The polynomial library is composed of various bricks: *(i)* the geometrical entities or convexes *(ii)* the L_2 orthonormal primal basis in which we express subsequently the polynomials, see [3, 4, 2], *(iii)* the definition and construction of point sets in convexes *e.g.* quadrature point sets and finally *(iv)* polynomials and finite elements.

As for the domain specific language *embedded*(DSEL) in C++, also known as FEEL++[1], it provides a very close syntax for numerical integration, projection and variational formulation to the corresponding mathematical language. To illustrate this, let us now consider the following system of Oseen equations

$$\alpha\mathbf{u} - \nu\Delta\mathbf{u} + (\boldsymbol{\beta}\cdot\nabla)\mathbf{u} + \nabla p = \mathbf{f}, \quad \text{in } \Omega \tag{1}$$

$$\nabla\cdot\mathbf{u} = 0, \quad \text{in } \Omega \tag{2}$$

$$\mathcal{B}(\mathbf{u}, p) = \mathbf{g}, \quad \text{on } \partial\Omega \tag{3}$$

where Ω is a bounded open polygonal domain in $\mathbb{R}^d, d = 2, 3$ with boundary $\partial\Omega$, $\mathbf{u}$ is the velocity of the fluid, p the pressure. The boundary conditions are represented by the operator $\mathcal{B}$. This system can reproduce the steady Stokes problem as well as fix point iterations for the Navier-Stokes case. As for the standard Navier-Stokes time discretizations, system (1)-(3) is still relevant after modifications of α, $\boldsymbol{\beta}$ and $\mathbf{f}$. An implementation in Life, once written in variational form, would be done as follows

```
AUTO( def, 0.5*(grad(v) + trans(grad(v))) );
AUTO( deft, 0.5*(gradt(u) + trans(gradt(u))) );
form2( Xh, Xh, M ) =
  integrate( elements(Xh->mesh()), IM,
      alpha*trans(idt(u))*id(v) + 2.0*nu*trace( trans(deft)*def )
      + trans(gradt(u)*idv(beta))*id(v) - div(v)*idt(p) + divt(u)*id(q) );
```

The first two lines allow to define the strain tensor for test and trial functions. Then we construct the bilinear form with its algebraic representation (matrix) `M` and where `Xh` is an instance of the function space data structure, `IM` is an instance of a quadrature method and finally we implement the expression of the form. In the context of fluid-structure interaction, it is often the case that the fluid flow is advection-dominated, stabilization

[1] Finite Element Embedded Language in C++

techniques are then required. We chose the *continuous interior penalty* (CIP) method to stabilize the velocity approximations, see [9]

$$j_\beta(\boldsymbol{u},\boldsymbol{v}) = \sum_{F\in\mathcal{F}_I}\int_F \left((\gamma_\beta + |\boldsymbol{\beta}\cdot\mathbf{n}|)\,\frac{h_F^2}{N^{3.5}}[\![\boldsymbol{\nabla}\mathbf{u}]\!]_F\cdot[\![\boldsymbol{\nabla}\mathbf{v}]\!]_F\right) ds \tag{4}$$

where $\mathcal{F}_I$ denotes the internal faces of the mesh, h_F denotes the length of the face F, N the order of the velocity approximation and γ_β the stabilization parameter. The notation $[\![\cdot]\!]_F$ denotes the jump of the quantity $\cdot$ across the face F, see [9] for the definition. An implementation of j_β reads as follows:

```
AUTO( stab_coeff , (γβ+abs(trans(N())*idv(beta)))*vf::pow(hFace(),2.0)/N^3.5 );
form2( Xh, Xh, M ) += integrate( internalfaces(Xh->mesh()), IM,
                                 stab_coeff * (trans(jumpt(gradt(u)))*jump(grad(v) ) ) );
```

In [11], the scaling for the stabilization term is $\gamma_\beta|\boldsymbol{\beta}\cdot\mathbf{n}|\frac{h_F^2}{N^{3.5}}$, allowing for more flexibility.

2. High order geometry

We now turn to a brief description of the high order geometry. We sketch the high order mesh construction and a versatile operator. We remark that using high order geometry impacts the quadrature rules and approximation order used in the variational forms and numerical integrals. Indeed, since we construct our polynomials in a reference element, to represent for example exactly order 5 polynomials in a real element of geometric order 4, we must use order 9 polynomials in the reference element.

2.1. Basic data structures

Denote Ω an open domain of $\mathbb{R}^d, d = 1,2,3$, $\mathcal{T}^{N_{\text{geo}}} = \cup_{k=1}^{N_{\text{el}}} T_k^{N_{\text{geo}}}$ its associated triangulation where $T_k^{N_{\text{geo}}} = \boldsymbol{\varphi}_k^{N_{\text{geo}}}(\hat{T})$, $\hat{T}$ is the reference element, and $\boldsymbol{\varphi}_k^{N_{\text{geo}}}$ is the geometric transformation of order N_{geo} that maps $\hat{T}$ to $T_k^{N_{\text{geo}}}$. Two strategies were followed *(i)* building $\boldsymbol{\varphi}_k^{N_{\text{geo}}}$ and the corresponding mesh following the steps in [3], and *(ii)* use the mesh generator [10] that provides high order meshes. Our implementation of the former allows for a priori better quality arbitrary order meshes but only in 2D while the latter allows for 2D and 3D up to order 5 but the interior points are not moved with respect to the element deformation, *e.g.* using Gordon-Hall transformations. Figure 1 plots the error $\pi - \int_\Omega 1\,dx$ where Ω is a radius one circle. The curved boundary of the circle is described with a mesh with order one to five generated by Gmsh.

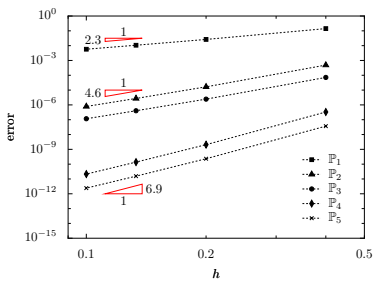


Figure 1: Convergence test for high order geometry

Finally we can introduce a few more standard notations. Let $\mathcal{T}_h^{N_{\text{geo}}}$ be a triangulation of the domain Ω, N_{geo} being the order of the elements of mesh and denote $h = \max_{T_k\in\mathcal{T}_h^{N_{\text{geo}}}} h_k$ ($h_k = \text{diam}(T_k)$). We define $\mathbb{P}_N(T_k)$ to be the space of polynomials of total degree N defined in $T_k \in \mathcal{T}_h^{N_{\text{geo}}}$. $\mathbb{P}_N(T_k)$ is associated with triangulations composed of simplices. $\mathbb{Q}_N(T_k)$

is the space of polynomials of degree N in each variable when T_k are tensorized convexes, *i.e.* quadrangles and hexahedra.

2.2. A useful operator

We now describe an operator which turns out to be useful in visualizing high order meshes and functions and in preconditioning matrices arising from high order discretizations. We denote

$$X_h^{N,N_{\text{geo}}}\left(\mathcal{T}^{N_{\text{geo}}}\right) = \left\{ v \in C^0(\mathcal{T}^{N_{\text{geo}}}), v_{|T_k^{N_{\text{geo}}}} \in \mathbb{P}_N(T_k^{N_{\text{geo}}}), T_k^{N_{\text{geo}}} = \boldsymbol{\varphi}_k^{N_{\text{geo}}}(\hat{T}) \in \mathcal{T}^{N_{\text{geo}}} \right\} \quad (5)$$

$X_h^{N,N_{\text{geo}}}$ is a function space which is spanned by nodal basis functions and the expansions are continuous. We would like to visualize not only high order functions of $X_h^{N,N_{\text{geo}}}$ without losing too much information but also high order meshes. To this end, we introduce the following interpolation operator

$$\Pi^{\mathbb{P}_1} : X_h^{N,N_{\text{geo}}} \mapsto X_{\tilde{h}}^{1,1} \quad (6)$$

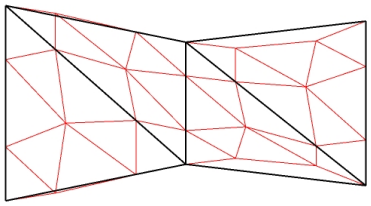


Figure 2: A first order mesh generated from a 4-th order mesh, In bold the first order mesh associated to the vertices of the elements of the 4-th order one.

where the mesh associated to $X_{\tilde{h}}^{1,1}$, a space spanned by a $\mathbb{P}_1$ Lagrange polynomials basis using a $\mathbb{P}_1$ geometric approximation, is constructed from the points associated to the degrees of freedom of $X_h^{N,N_{\text{geo}}}$. We remark that the points on $\partial\mathcal{T}^1$ are located, thanks to $\boldsymbol{\varphi}^{N_{\text{geo}}}$, exactly on $\partial\mathcal{T}^{N_{\text{geo}}}$ thus we retain a good approximation of the boundary. The construction of $\mathcal{T}_{\tilde{h}}^1$ uses the following ingredients *(i)* $\boldsymbol{\varphi}^{N_{\text{geo}}}$, *(ii)* the degrees of freedom table of $X_h^{N,N_{\text{geo}}}$ and *(iii)* $\hat{\mathcal{T}}^1$ a mesh of $\hat{T}$ whose vertices are the points associated to the degrees of freedom in the reference element $\hat{T}$. Figure 2 displays the results of the algorithm on a 4-th order mesh of a 3-rd order boundary domain. We indeed observe that the points on the boundary edges lie on the fourth order boundary.

As to using this operator to build preconditioners for matrices arising from high order discretization, this comes from the features *(i)* $\dim X_{\tilde{h}}^{1,1} = \dim X_h^{N,N_{\text{geo}}}$ and *(ii)* if $v \in X_h^{N,N_{\text{geo}}}$, $\Pi^{\mathbb{P}_1}(v)$ has the same nodal values on the mesh associated to $X_{\tilde{h}}^{1,1}$ as v. Its numerical features are discussed in [3].

3. High order structure

Regarding the structure solver, prescribed displacement, StVenant-Kirchhoff and the generalized string models have been implemented. The former is used in section 5.3 while the later is being used in the fluid-structure section 5.4. The associated equations to the generalized string models are stated in the solid reference domain $\hat{\Omega}_s = \{(r, z) : r =$

R_0, $z \in [0, L]\}$. σ_Σ is the radial component of the stress vector of the fluid acting on the structure, see [6, 7]. The displacement d satisfies the following equation

$$\rho_s h \frac{\partial^2 d}{\partial t^2} - kGh \frac{\partial^2 d}{\partial z^2} + \frac{Eh}{1-\nu^2} \frac{d}{R_0^2} - \gamma \frac{\partial^3 d}{\partial z^2 \partial t} = \sigma_\Sigma \quad \text{on } \hat{\Omega}_s \tag{7}$$

and homogeneous Dirichlet boundary conditions. Here h is the wall thickness, k is the Timoschenko shear correction factor, G the shear modulus, E the Young modulus, ν the Poisson ratio, ρ_s the wall density, γ a viscoelastic parameter. The strategy to solve (7) is *(i)* BDF_n for the time discretization introducing the first derivative of the displacement as an additional unknown and *(ii)* high order discretization in space.

4. High order Navier-Stokes

4.1. Algebraic framework

We consider now the system (1)-(3). The finite element spaces setting for the velocity and pressure fields is the following:

$$\boldsymbol{V}_N = \{\mathbf{v} \in C^0\left(\overline{\Omega}\right)^2 : \ \mathbf{v}_{|T_k} \in \mathbb{P}_N^2(T_k), \ \forall T_k \in \mathcal{T}_h^{N_{\text{geo}}}\} \tag{8}$$

and

$$Q_N = \{q \in C^0\left(\overline{\Omega}\right) : \ q_{|T_k} \in \mathbb{P}_M(T_k), \ \forall T_k \in \mathcal{T}_h^{N_{\text{geo}}}\} \tag{9}$$

where $M = N - 1$ or $M = N - 2$. When using these discretization spaces, we shall refer to the $\mathbb{P}_N$ - $\mathbb{P}_M$ method. The discontinuous pressure version of these methods are also considered and denoted by $\mathbb{P}_N$ - $\mathbb{P}_M^{disc}$. In what follows, the spaces are spanned by Lagrange bases constructed at the Fekete points [3] associated to the type of convex. We shall display only a few results to support our choices. For an exhaustive presentation of the various results obtained, see [11]. The discretization of the system (1)-(3) using these methods leads to the linear system

$$\begin{bmatrix} F_N & G_N \\ D_N & 0 \end{bmatrix} \begin{bmatrix} \mathbf{U} \\ \mathbf{P} \end{bmatrix} = \begin{bmatrix} \mathbf{F}_1 \\ \mathbf{F}_2 \end{bmatrix}. \tag{10}$$

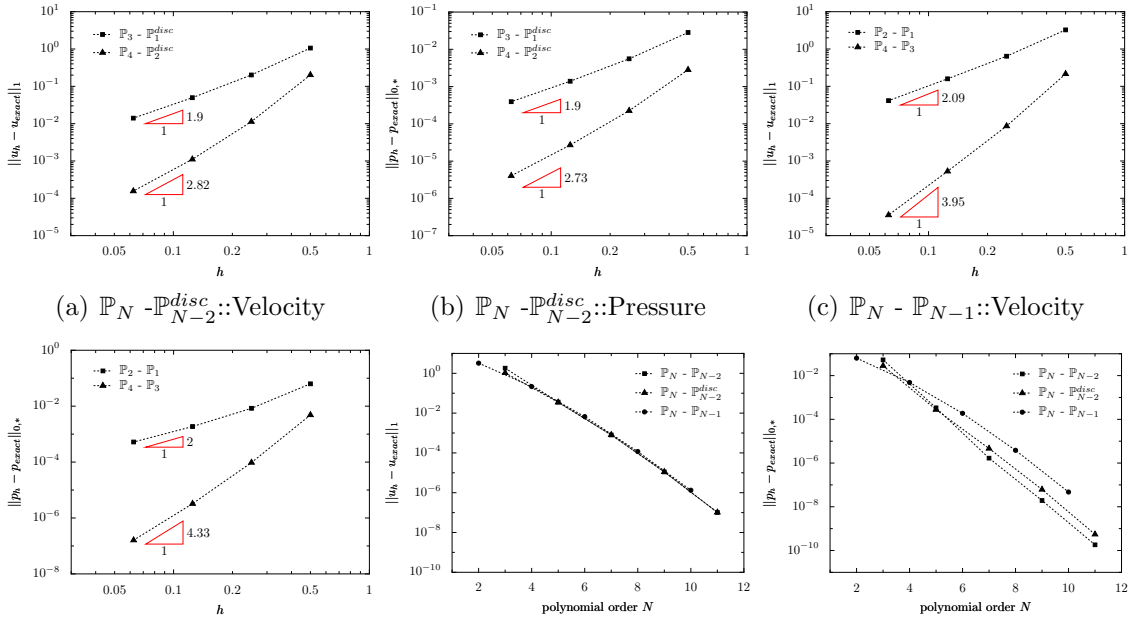
where *(i)* $F_N = \alpha M_N + \nu H_N + C_N$ with M_N, H_N and C_N corresponding to the discretization of the identity, Laplace and convection operators respectively and *(ii)* G_N and D_N correspond to the discretization of the gradient and divergence operators respectively.

In order to solve (10), the following preconditioning strategy has been used in all the Navier-Stokes related results: *(i)* we build a complete LU factorization which does not break since we set the pressure weakly either at inflow or at outflow, *(ii)* we reuse this LU factorization as preconditioner in combination with a GMRES solver in all subsequent time steps — we note that initially the number of linear iterations is 1 — *(iii)* we rebuild the LU factorization when the number of the GMRES iterations reaches say 10 or 20 iterations. It is very interesting to observe that the preconditioner is rebuilt very few times during the whole computation and displays far better performances than an ILU approaches where the

right parameters need to be found, the cost of the LU factorization is in fact marginal when compared to the overall cost. In 2D we have found this strategy extremely effective and robust with respect to h and N. In the case of internal flows, Life allows to easily augment the approximation space and add Lagrange multipliers to enforce a particular constraint on the pressure, *e.g.* zero mean pressure, to ensure uniqueness then our strategy still holds.

4.2. Choice of the discretization spaces

To help our choices, we consider the Kovasznay solution of the steady Stokes equations (see page 177 of [5]) and we use the $\boldsymbol{H}^1$ norm of the error on the velocity and the norm $\|q\|_{0,*} = \|q - m_\Omega(q)\|_0$, where $m_\Omega(q)$ denotes the average of q in Ω, for the pressure to compare the accuracy of the different methods. In the results displayed in this section N_{geo} was set to 1.



(a) $\mathbb{P}_N$ -$\mathbb{P}_{N-2}^{disc}$::Velocity (b) $\mathbb{P}_N$ -$\mathbb{P}_{N-2}^{disc}$::Pressure (c) $\mathbb{P}_N$ - $\mathbb{P}_{N-1}$::Velocity

(d) $\mathbb{P}_N$ - $\mathbb{P}_{N-1}$::Pressure (e) N Convergence::Velocity (f) N Convergence::Pressure

Figure 3: Error plots for the velocity and pressure for the various element types.

First, we compare the different methods of the preceding section by fixing N and varying h. The $\mathbb{Q}_N$ - $\mathbb{Q}_{N-2}^{disc}$ method is widely used, a quasi-optimal error estimate is obtained in the quadrangular mesh case and we recover similar results in the triangular one, see Figures 3(a) and 3(b). Although the velocity is approximated using order N basis functions, the error decays only as h^{N-1}, for $N = 3, 4$. As for the pressure, the error behaves optimally as h^{N-1}. Similar results are obtained for $\mathbb{P}_N$ - $\mathbb{P}_{N-2}$. The $\mathbb{P}_N$ - $\mathbb{P}_{N-1}$ methods were studied by Brezzi and Falk, they are also known as the Taylor-Hood element. The results displayed on Figures 3(c) and 3(d) show an optimal convergence behavior as h decreases for both pressure and velocity. We now fix h and vary N. The most accurate method is the $\mathbb{P}_N$ - $\mathbb{P}_{N-2}$ one, see Figures 3(e) and 3(f).

5. Putting it all together: Fluid structure interaction

We now consider the unsteady Navier-Stokes equations in a domain with a moving boundary. We use the Arbitrary Lagrangian Eulerian (ALE) framework to keep track of the domain's deformation, see e.g. [6].

5.1. Formulation

The system's evolution is studied in the interval $I = [t_0, T]$ and we denote Ω_0 the reference configuration for instance the domain at time $t = t_0$. The position of a point in the current domain Ω_t is denoted by $\mathbf{x}$ (in the Eulerian coordinate system) and by $\mathbf{Y}$ in the reference domain Ω_0. We consider now the simple 2D model depicted on figure 4 where $\Omega_0^{\mathcal{D}}$ is fixed and Ω_0^{σ} evolves according to some viscoelastic model. We introduce the so called ALE map, a family of mappings

$$\mathcal{A}_t : \Omega_0 \mapsto \Omega_t, \quad \mathbf{Y} \longrightarrow \mathbf{x}(\mathbf{Y}, t), \quad t \in [t_0, T]. \tag{11}$$

and the domain's deformation velocity, denoted $\mathbf{w}$, given by $\mathbf{w}(\mathbf{x}, t) = \frac{\partial \mathcal{A}_t}{\partial t}\Big|_{\mathbf{Y}}$. The Navier-Stokes equations now read in the ALE framework

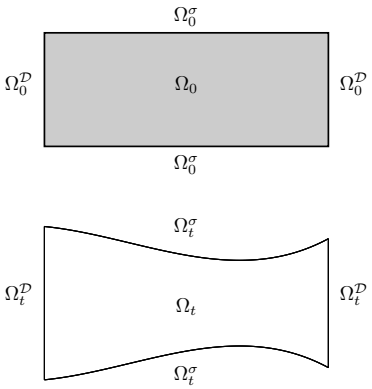


Figure 4: Reference domain at time $t = t_0$ (top) and current domain at time t (bottom).

$$\left.\frac{\partial \mathbf{u}}{\partial t}\right|_{\mathbf{Y}} + [(\mathbf{u} - \mathbf{w}) \cdot \boldsymbol{\nabla}]\mathbf{u} + \nabla p - 2\nu \mathbf{D}(\mathbf{u}) = \mathbf{f}, \quad \boldsymbol{\nabla} \cdot \mathbf{u} = 0 \tag{12}$$

in Ω_t, for all $t \in I$. $\frac{\partial \mathbf{u}}{\partial t}\Big|_{\mathbf{Y}}$ denotes the time derivative in the ALE framework and $\mathbf{D}(\mathbf{u}) = \frac{1}{2}\left(\boldsymbol{\nabla}\mathbf{u} + \boldsymbol{\nabla}\mathbf{u}^T\right)$ is the strain tensor. A Dirichlet or Neumann boundary condition is imposed at the inflow, homogeneous Neumann at the outflow and the domain's deformation velocity on Ω_t^{σ}.

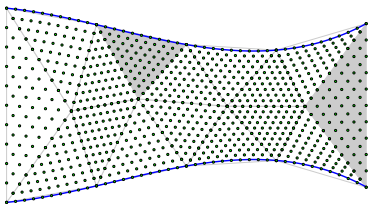


Figure 5: Effect of $\mathbf{x}_h^3$ on a equidistributed point set in the whole mesh.

5.2. Arbitrary Lagrangian Eulerian map

Assume that $\Omega_t^{\mathcal{D}}$ is described in terms of polynomials of degree N_{geo} and we denote $\mathcal{T}_{0,h}^{N_{\text{geo}}}$ a triangulation of the domain $\Omega_{0,h} \approx \Omega_0$. The algorithm runs as follows : *(i)* we construct the harmonic extension of the boundary data denoted $\mathbf{x}_h^1$ — other types of extension can be found in [6, 8], — *(ii)* we project $\mathbf{x}_h^1$ onto the space $X_h^{N_{\text{geo}},1}$ and we denote $\mathbf{x}_h^{N_{\text{geo}}}$ the projection and *(iii)* we update the values of the degrees of freedom of $\mathbf{x}_h^{N_{\text{geo}}}$ at the boundary edges and at the interior of the boundary elements in the reference mesh so that the image of $\Omega_{0,h}$ is conform nodally with the high order description of the boundary. We emphasize here that the procedure keeps the internal elements with straight edges, this allows to differentiate internal elements and the high order boundary elements and use $\mathbb{P}_1$ geometry approximation and lower order quadratures in the internal elements.

5.3. Navier-Stokes ALE solver with a prescribed boundary movement

We are now ready to assemble the various building blocks described previously and build our ALE Navier-Stokes solver. Consider $\Omega_0 = (0,5) \times (-1,1)$, in our test Ω_t is obtained from Ω_0 by applying the following displacement law

$$\boldsymbol{d}(x,t) = \begin{cases} 0, & 0 \leqslant t \leqslant 1 \\ 0.08x(5-x)f(t), & 1 < t < 3 \\ 0.08x(5-x), & t \geqslant 3 \end{cases} \tag{13}$$

with $t \in I = [0;5]$ and $f(t) = 2.5(t-1)^2(0.3 - 0.1(t-1))$

The discretization of the differential equations is done using a non-conservative scheme (see page 82 of [6] for more details). The time derivatives are discretized using first to third order BDF schemes. The mesh velocity is calculated also with a BDF scheme of the same order as for the velocity time derivative and the nonlinear convective term is linearized with an extrapolation formula also of the same order. We check the convergence order of our solution methods for (12) by considering the solution of a Poiseuille flow, say $(\mathbf{u}_{Poiseuille}, p_{Poiseuille})$ that solves the steady Navier-Stokes equations in the reference domain. $\mathbf{u}_{Poiseuille}$ is prescribed on Ω_t^σ as boundary condition which is the solution that we must recover in the domain. In Figure 6, the error $||e_h|| = \left(\Delta t \sum_{n=0}^{T_N} ||\mathbf{u}_h^n - \mathbf{u}^n_{\text{Poiseuille}}||^2_{L^2(\Omega_t)} + ||p_h^n - p^n_{\text{Poiseuille}}||^2_{L^2(\Omega_t)}\right)^{1/2}$ is plotted as a function of Δt. T_N is the number of subintervals in which $[0,5]$ is discretized for the different BDF schemes. The exact solution was taken as $\mathbf{u}_{Poiseuille} = y(1-y)\mathbf{e}_1$, $\nu = 10^{-3}$ and $p_{Poiseuille} = -2\nu(x-5)$. Note that in this simulation, the stabilization term (4) is *not* used. The same results are obtained using $\mathbb{P}_4 - \mathbb{P}_2$ method and a $\mathbb{P}_2$ geometry.

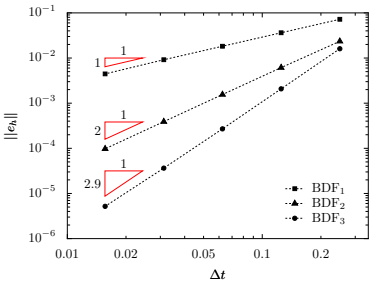
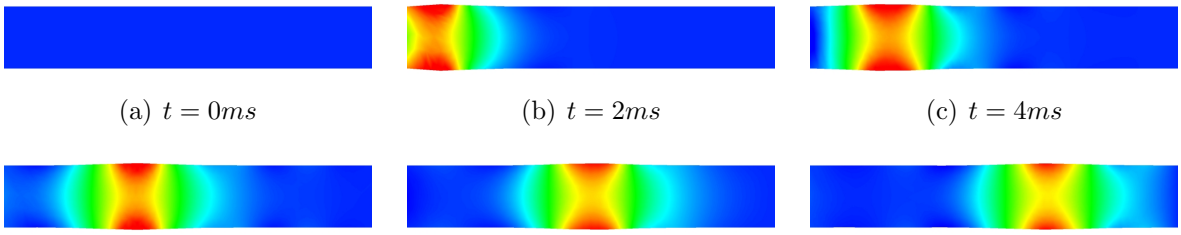


Figure 6: $\|e_h\|$ for different time integration schemes using a $\mathbb{P}_6 - \mathbb{P}_4$ element and a $\mathbb{P}_4$ geometric description of the domain.

5.4. Fluid-Structure interaction

The methodology we adopt to solve the coupled fluid-structure interaction problem is to perform standard fix point iterations alternating the fluid and the structure solvers. The algorithm reads as follows: for each t_n

1. Extrapolate the structure displacement: $d^{n+1}_{(0)} = d^n + \Delta t \dot{d}^n$, $\dot{d}^{n+1}_{(0)} = \dot{d}^n$
2. for $j = 1, \ldots$ (fix point iterations)
   (a) given $d^{n+1}_{(j-1)}$, calculate the ALE map and update the computational domain
   (b) solve the Navier-Stokes equations
   (c) calculate the shear stress at the moving boundary of the fluid
   (d) solve for the structure displacement, $d^{n+1,*}$ (and $\dot{d}^{n+1,*}$)
   (e) if $\max\{\frac{||d^{n+1,*} - d^{n+1}_{(j-1)}||_{L_2}}{||d^{n+1,*}||_{L_2}}, \frac{||\dot{d}^{n+1,*} - \dot{d}^{n+1}_{(j-1)}||_{L_2}}{||\dot{d}^{n+1,*}||_{L_2}}\} < tol$, then advance for the next timestep with $d^{n+1} = d^{n+1,*}$ and $\dot{d}^{n+1} = \dot{d}^{n+1,*}$; otherwise (Aitken relaxation) $d^{n+1}_{(j)} = \theta d^{n+1,*} + (1-\theta) d^{n+1}_{(j-1)}$ and $\dot{d}^{n+1}_{(j)} = \theta \dot{d}^{n+1,*} + (1-\theta)\dot{d}^{n+1}_{(j-1)}$



(a) $t = 0ms$ (b) $t = 2ms$ (c) $t = 4ms$

(d) $t = 6ms$ (e) $t = 8ms$ (f) $t = 10ms$

Figure 7: Pressure pulse propagating through the pipe. Fluid discretized with BDF_1 and $\mathbb{P}_6 - \mathbb{P}_4$ elements, $h = 0.5$, $\Delta t = 10^{-4}$, no stabilization and $tol = 10^{-6}$. For the structure we used BDF_2 and $\mathbb{P}_2$ elements.

We first note that the fix point method to solve the fluid-structure interaction (FSI) problem is a naive approach. Better alternatives exist in the literature, but this paper focuses on the high order finite element/geometry part, the time discretization of each subproblem and putting them together. Second, the shear stress is explicitly calculated in the boundary of the fluid domain and a nodal projection is used to pass this quantity to the structure solver. Third, we implemented an Aitken acceleration procedure to determine the relaxation parameter θ at each iteration (see [7] for more details). Finally, the meshes of the structure and the fluid are nodally conform.

To check the strategy, we consider a test case as in [6] (see page 143). This example uses physiological parameters for the FSI model in the context of haemodynamics. We changed the original problem only by imposing a Dirichlet velocity profile at the inlet given by:

$$\begin{aligned}\boldsymbol{u}(0, y, t) = 343.99(0.25 - y)^2(-1357t^9 + 7443t^8 - 17099t^7 + 21255t^6 \\ -15356t^5 + 6379t^4 - 1368t^3 + 97t^2 + 6t)\boldsymbol{e}_1, \text{ with } \boldsymbol{e}_1 = (1, 0)^T.\end{aligned}$$

In Figure 7 we plot the pressure field associated with this inlet profile at several time steps. We can the pressure wave propagating through the pipe due to the elastic behavior of the walls. We performed successfully several simulations, with different time/space discretizations, namely, $\mathbb{P}_N - \mathbb{P}_{N-2}$ $(N = 3, 4, 6)$ finite elements for the fluid, $\mathbb{P}_1$ geometry and BDF_n, $n = 1, 2, 3$ as time integrator. The structure model was discretized with $\mathbb{P}_N$, $N = 1, 2, 3$ elements. The current results using higher order geometrical transformation for the fluid exhibit instabilities that we are currently addressing.

6. Conclusion

In this paper we presented a non standard approach to solve FSI problems. Our method uses several complex tools that, to our knowledge, are being used together for the first time in this context (high order space/time discretization, geometrical transformation and CIP stabilization). Our current results confirm the expected accuracy properties of the ALE Navier-Stokes solver. The FSI solver was successfully used to simulate a haemodynamics problem with realistic data. Our next step is to use higher order geometrical elements for the fluid in the FSI context.

.

Acknowledgements

The authors would like to thank Prof. A. Quarteroni, Dr. S. Deparis and A. Quaini from EPFL and Prof. E. Burman from Univ. of Sussex for the fruitful discussions we had. The first author was supported by Fundação para a Ciência e Tecnologia through grant SFRH/BD/22243/2005 of POCI2010/FEDER.

References

[1] C. Prud'homme, A domain specific embedded language in C++ for automatic differentiation, projection, integration and variational formulations, Scientific Programming 14 (2) (2006) 81–110.

[2] C. Prud'homme, Life: Overview of a unified C++ implementation of the finite and spectral element methods in 1D, 2D and 3D, in: Workshop On State-Of-The-Art In Scientific And Parallel Computing, Lecture Notes in Computer Science, Springer-Verlag, 2006, p. 10, accepted.

[3] G. E. Karniadakis, S. J. Sherwin, Spectral/hp element methods for computational fluid dynamics, 2nd Edition, Oxford Universtity Press, Oxford, 2004.

[4] R. C. Kirby, Algorithm 839: Fiat - a new paradigm for computing finite element basis functions, ACM Trans. Math. Software 30 (4) (2004) 502–516.

[5] C. Canuto, M. Y. Hussani, A. Quarteroni, T. A. Zang, Spectral Methods: Evolution to Complex Geometries and Applications to Fluid Dynamics, Springer-Verlag, New York and Berlin, 2006.

[6] F. Nobile, Numerical approximation of fluid-structure interaction problems with application to haemodynamics, Ph.D. thesis, EPF Lausanne (2001).

[7] S. Deparis, Numerical analysis of axisymmetric flows and methods for fluid-structure interaction arising in blood flow simulation, Ph.D. thesis, EPF Lausanne (2004).

[8] R. Bouffanais, Simulation of shear-driven flows: transition with a free surface and confined turbulence, Ph.D. thesis, EPF Lausanne (2007).

[9] E. Burman, M. Fernandez, Continuous interior penalty finite element method for the time-dependent Navier-Stokes equations: space discretization and convergence, Numer. Math. 107 (2007), no. 1, 39–77.

[10] C. Geuzaine and J.-F. Remacle, Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities, Submitted to the International Journal for Numerical Methods in Engineering, 2008

[11] G. Pena, High order methods in space and time for the Navier-Stokes equations in a moving domain and applications, Ph.D. thesis (in preparation), EPF Lausanne (2009)