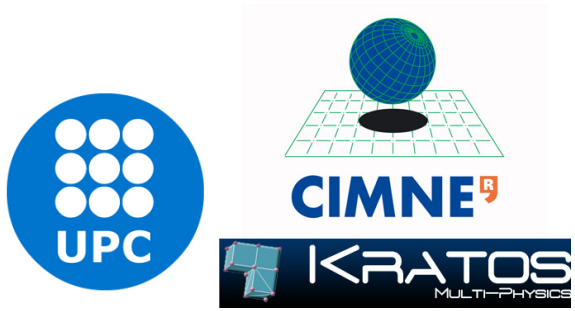
The Finite Element Method for Fluid-Structure Interaction with open source software

Embedded methods for Navier Stokes

Riccardo Rossi Pooyan Dadvand ....

Kratos Team

Centre Internacional de Mètodes Numèrics en Enginyeria (CIMNE)



September 16, 2014

Outline

Traditional, Constrained Mesh approach

"Body fitted meshes" are the traditional way of handling fluid problems.

- match exactly the surface discretization
- simplicial meshes can be constructed automatically
- can be anisotropically graded to provide greater resolutions in boundary layers
- **Provide maximal accuracy**

remark 1

Difficult to construct robustly. Existing techniques are Constrained Delaunay, Advancing front

remark 2

not (easily) parallel

non-body fitted meshes

Non body fitted meshes **do not attempt matching exactly the surface discretization**

- One possibility is to construct volume meshes that directly approximate the surface to be represented without representing it exactly → Much more robust and faster than the first alternative as it allows pathologic cases
- Second possibility is to identify how the surface cuts the mesh and modify the solver to deal with this → this is the subject of my talk

can we cut elements?

A first obvious question is ...can we cut elements? the answer is YES BUT...

- no guarantee on the quality of the mesh obtained (worst quality has consequences on accuracy, speed, robustness, etc.)
- need to do the cut with some care to mantain continuity across cut elements (alternatively hanging nodes must be used)
- time step is reduced for explicit methods
- FE graph is changed
- shape function gradients may be come very steep

doing constrained refinement

The idea is essentially an extension of the former. It is possible to use the same techniques used in mesh generation to do refinement.
May work but:

- requires having control of a mesher (!)
- difficult to have it robust (As difficult or more than doing mesh generation from scratch)
- again ... not parallel

IDEA: Immersed mesh approaches

A rather old idea is that of "Immersed approaches". The essential ingredient is to have a discretization of a structure and to know where the structure falls within the Fluid domain
algorithmically it proceeds as follows (very naive description)

1. evolve the fluid in time as if the structure was not reacting to such evolution
2. map the fluid velocity field onto the structure and deform the structure according to the predicted fluid velocity
3. The structure is not in equilibrium: for each node we can compute the residual of the momentum equation, which is a measure of the forces not correctly balanced
4. apply such forces onto the fluid field and compute fluid velocity taking them into account
5. go back to step (ii) and repeat until convergence

IDEA: Immersed mesh approaches

unfortunately the devil is in the details ... admittedly i am not the best expert in immersed methods...however...

1. how to transfer structural forces to the fluid domain?
2. what if mesh distribution is very different from the two domains? (ok if structure is fine and fluid coarse...what about the opposite??)
3. Sharp gradients of velocity/pressure may appear in the solution. This typically leads to oscillations
4. not very accurate (order of convergence??)
5. can not model completely rigid structures

embedded meshes approaches

The essential idea is that one part of the fluid domain is covered by the solid domain and is "turned off" accordingly.
Surface coupling is imposed by enforcing a dirichlet boundary condition on velocity on the modified fluid domain.
The structural shape is *approximately* represented on the fluid domain. "Fine details" which can not be represented by the fluid discretization are discarded

How to represent the structural domain onto the fluid

The way we employ is through a distance function. Continuous distance function are appropriate for solids within a fluid.

Solution

Use a discontinuous distance function

Remark

any element can ONLY REPRESENT A SINGLE CUTTING PLANE

Continuous distance function are appropriate for solids within a fluid. Unfortunately they are not suitable to handle membranes (no domain for extrapolation!).
and...Membranes are needed due to our project with TUM (uLites)

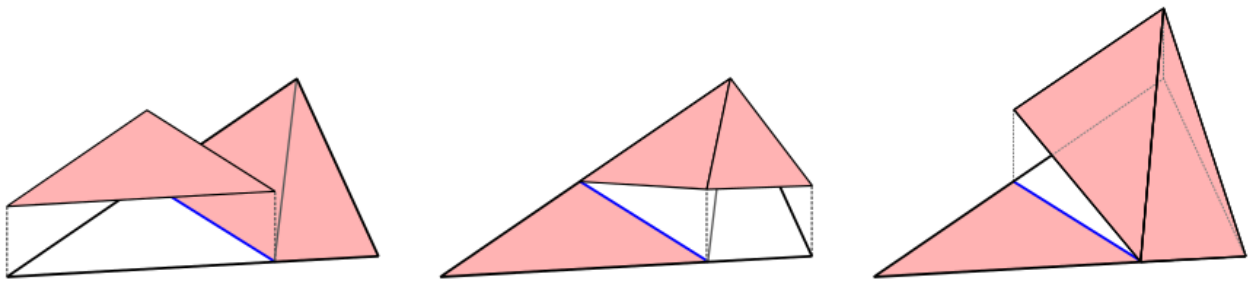
first option: extrapolating and fixing the velocity field

A first option, suitbale for solid objects, is to extrapolate the fluid velocity field and to fix the velocities within the solid object to a value that guarantees that the correct value is found at the solid's boundary.
Only possible if enough nodes exist within the object. See the work of Codina and Baiges for details.
The point is...how to impose the boundary conditions within your system?

substituting shape functions

Once the cut is actually identified we need to modify the solution space so **to make it suitable to represent the solution**.
Easiest possibility is to make it **DISCONTINUOUS** and to *substitute* the FE shape function by suitably defined discontinuous shape functions

R.F. Ausas et al. / Comput. Methods Appl. Mech. Engrg. 199 (2010) 1019–1031



this shape functions (first proposed in 2010 by Ausas and Buscaglia) are based on the idea that shape functions are constant from the node to the position of the discontinuity on each cut edge

how to actually compute the new shape functions

begin by splitting the element so that the first 4 nodes (3 in 2d are the original nodes) and the following are edge nodes (here we indicate with N the shape functions)

```
set to zero gradients and shape functions
loop on subelements
   compute Area, Local_Gradients, N=1/nnodes, subel_sign

   loop on nodes
       if(node_sign != subel_sign) N(node) = 0

   loop on the edge nodes in the subelement
       get nodes at the end of the corresponding edge (I and J)
       if(sign(I) == subel_sign)
           N[i] += N[edge_node], grad_I += grad(edge_node)
       else
           N[J] += N[edge_node], grad_J += grad(edge_node)

(verify that N and grad sum to 1)
```

... now integrate as usually with gauss points on every subelement and solve the final system

handling slip boundary condition

There exist different possibilities in handling slip boundary conditions. The way we propose is by imposing the slip condition in a variational sense
that is, by asking that $\int_{\Gamma_{slip}} \vec{u} \cdot \vec{n} = 0$. This is **AUTOMATIC** if we integrate by parts the divergence condition and impose the pressures on the outflow boundary.

solid objects

For solid objects, it is enough to fix to zero the velocity and pressures on the interior nodes. Since the velocities within the fluid are independently computed and the same for the pressure, the space is suitable to represent correctly the flux within the fluid. This way it is AUTOMATIC to apply slip boundary conditions on the object boundary.

membrane-like objects

The enriched shape functions make the fluid on the two sides of the membrane to behave as if part of two separated domains. in this sense it is completely automatic to have completely distinct behaviours on the two sides.

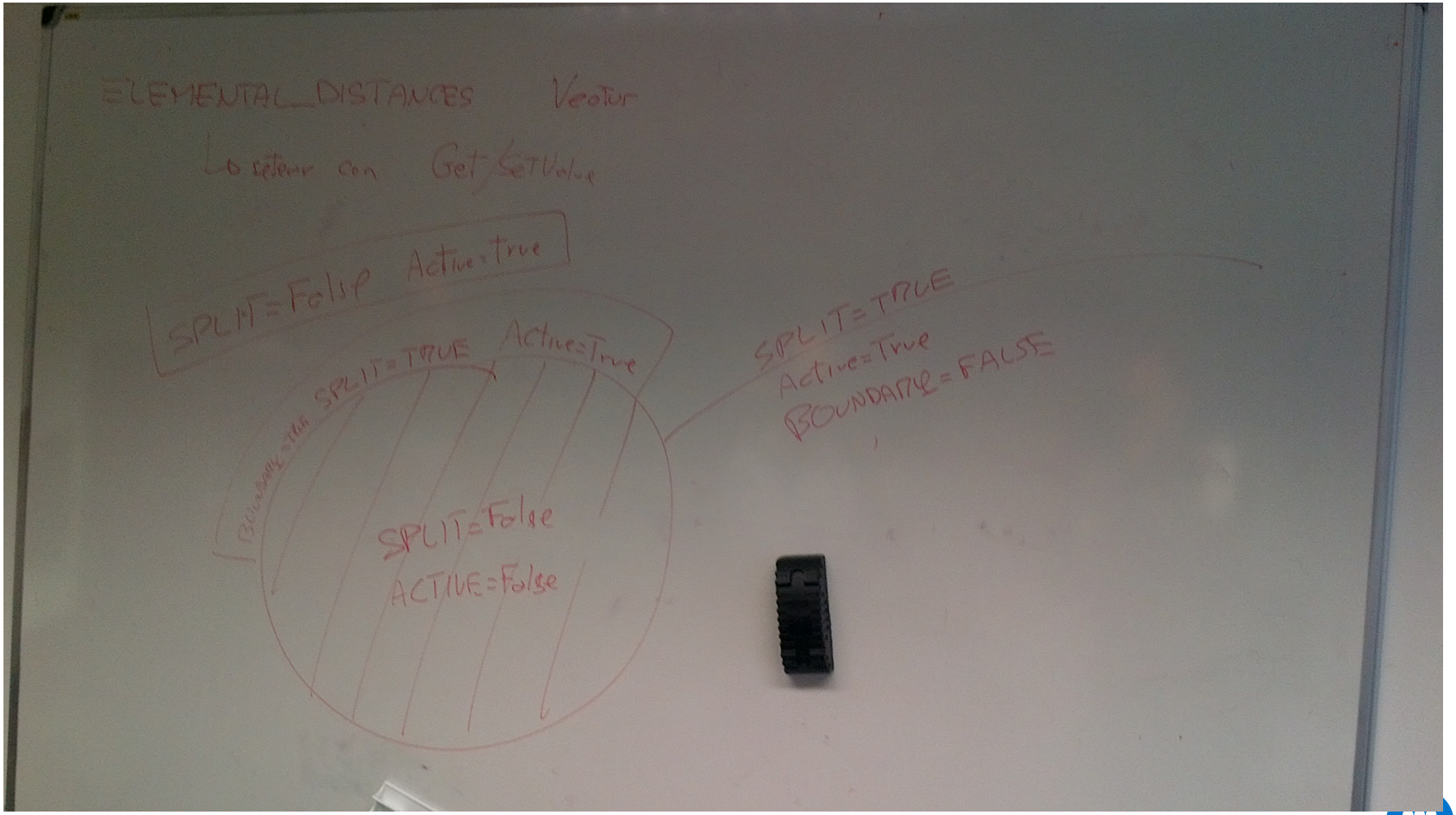
membrane-like objects

a tent like structure:

- velocity view
- pressure view
- vectors

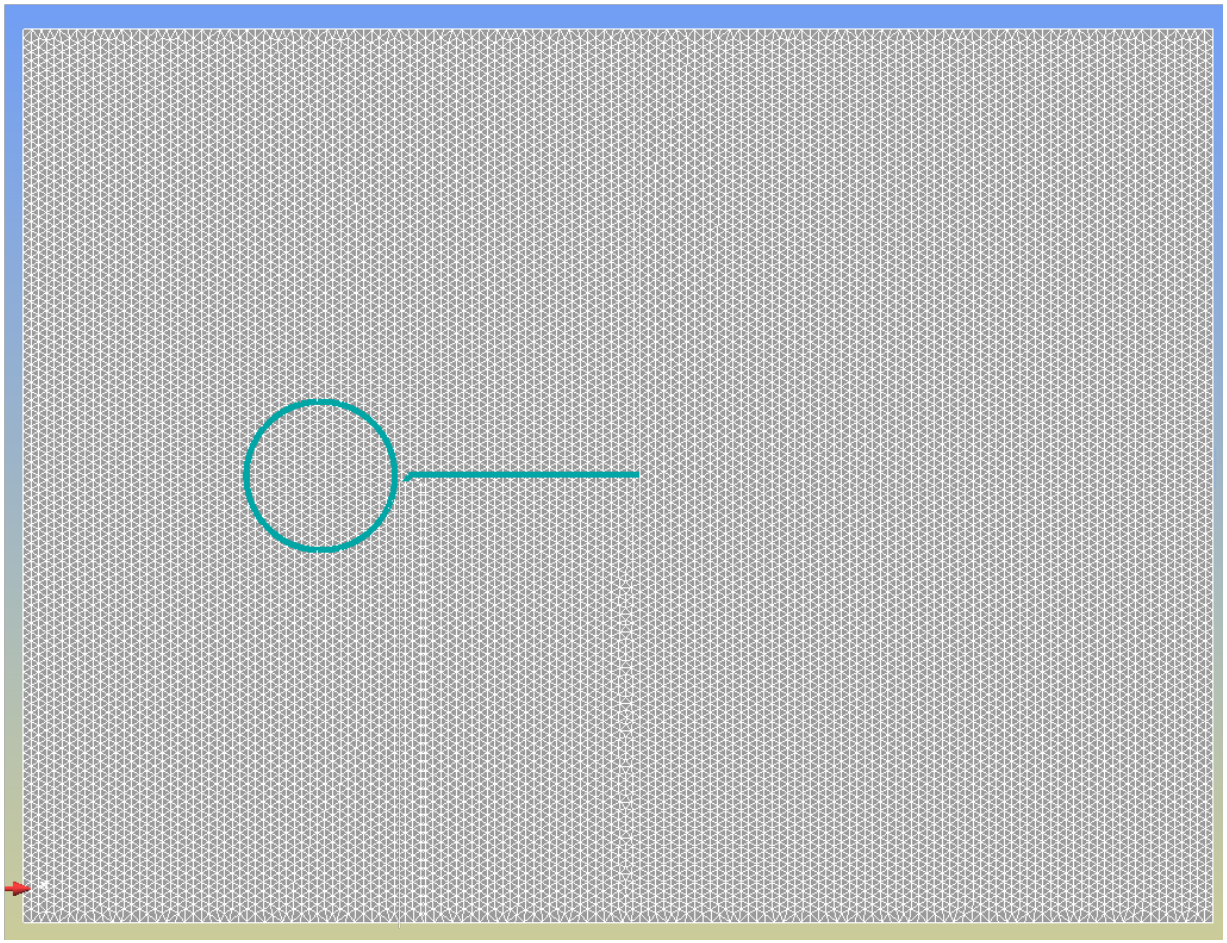
mixing solids and membranes

The (arbitrary) convention we follow. (we wish to use flags)



mixing solids and membranes

"Geometry definition"



mixing solids and membranes

some videos...

- velocity view
- pressure view
- vectors

very nice ... but...is there a BUT?

In Ausas approach we **substitute** the finite element space in cut elements by a modified one, **WITH DIFFERENT APPROXIMATION PROPERTIES**. Main feature is that it embeds a discontinuity and that it has **normal derivatives forced to zero**.

Problem:

approximation properties get worst $\longrightarrow$ Convergence drops from h^2 to $h^{3/2}$

going further

The solution is to **enrich** the finite element space instead of **substituting** it. This implies ADDING new shape functions to the solution space. With this approach *any solution that can be represented in the finite element space can also be represented in the new space. But the new space can also represent new features of the solution, which are exactly the thing it is designed for.*

requirements for the new space

For FSI problems we need both velocity and pressures to have

- both pressure and velocity should be allowed to be discontinuous across the interface
- gradient of pressure and gradient of velocity also discontinuous
- shape functions should be purely element based, that is, based on variables that are local to each element

The third point is very important for implicit methods: If enrichment functions are not local to elements, enrichment variables are much like normal finite element functions. This implies that if the position of the structure is changed the graph of the system matrix changes (like in remeshing) and the number of total variables increases. Local functions on the contrary can be condensed statically, without changing the shape function nor increasing the solve cost.

remark:

Should take care of variational crimes! Pressure lives in L^2 and hence can be discontinuous. Velocity lives in H^1 and hence can not be discontinuous. If we do not take care of this, we may get into problems

requirements for the new space

For FSI problems we need both velocity and pressures to have

- both pressure and velocity should be allowed to be discontinuous across the interface
- gradient of pressure and gradient of velocity also discontinuous
- shape functions should be purely element based, that is, based on variables that are local to each element

The third point is very important for implicit methods: If enrichment functions are not local to elements, enrichment variables are much like normal finite element functions. This implies that if the position of the structure is changed the graph of the system matrix changes (like in remeshing) and the number of total variables increases. Local functions on the contrary can be condensed statically, without changing the shape function nor increasing the solve cost.

remark:

Should take care of variational crimes! Pressure lives in L^2 and hence can be discontinuous. Velocity lives in H^1 and hence can not be discontinuous. If we do not take care of this, we MAY get into problems (but the problems MAY be minor)

Let's focus on the laplacian case (Problem of temperature)

remark:

Temperature lives in H^1 , we WILL commit a variational crime in the following!

The idea is to add, within each element, two new shape functions N^* and N^{**} the first representing a discontinuity in the gradient, the second representing a discontinuity in the function. This implies that the variable of interest will be described as
$T = \sum N_I T_I + N^* T^* + N^{**} T^{**}$
where T_I are the nodal finite element values and T^* T^{**} are the new enrichmement variables. Each element have T^* T^{**} completely independent on the neighbours!!!!

The formulation

Original problem to be solved $\Delta T = 0$. Applying Galerkin we get: $\int_\Omega w \Delta T d\Omega = 0$ and integrating by parts $-\int_\Omega \vec{\nabla} w \cdot \vec{\nabla} T + \int_\Omega w \vec{\nabla} T \cdot \vec{n} = 0$
The corresponding LHS becomes thus

$$\begin{pmatrix} \vec{\nabla}N_0\vec{\nabla}N_0 & \vec{\nabla}N_0\vec{\nabla}N_1 & \vec{\nabla}N_0\vec{\nabla}N_2 & \vec{\nabla}N_0\vec{\nabla}N^* & \vec{\nabla}N_0\vec{\nabla}N^{**} \\ \vec{\nabla}N_1\vec{\nabla}N_0 & \vec{\nabla}N_1\vec{\nabla}N_1 & \vec{\nabla}N_1\vec{\nabla}N_2 & \vec{\nabla}N_1\vec{\nabla}N^* & \vec{\nabla}N_1\vec{\nabla}N^{**} \\ \vec{\nabla}N_2\vec{\nabla}N_0 & \vec{\nabla}N_2\vec{\nabla}N_1 & \vec{\nabla}N_2\vec{\nabla}N_2 & \vec{\nabla}N_2\vec{\nabla}N^* & \vec{\nabla}N_2\vec{\nabla}N^{**} \\ \vec{\nabla}N^*\vec{\nabla}N_0 & \vec{\nabla}N^*\vec{\nabla}N_1 & \vec{\nabla}N^*\vec{\nabla}N_2 & \vec{\nabla}N^*\vec{\nabla}N^* & \vec{\nabla}N^*\vec{\nabla}N^{**} \\ \vec{\nabla}N^{**}\vec{\nabla}N_0 & \vec{\nabla}N^{**}\vec{\nabla}N_1 & \vec{\nabla}N^{**}\vec{\nabla}N_2 & \vec{\nabla}N^{**}\vec{\nabla}N^* & \vec{\nabla}N^{**}\vec{\nabla}N^{**} \end{pmatrix} \quad (1)$$

the trick is that the last two rows and columns are independent element by element and can thus condensed statically **prior to building!!**

one simple example

(some videos)

- temperature
- temperature contours

final comment:

All of these methods work for sufficient fine meshes. Important to be able to refine dynamically the mesh where you like it
we follow the procedure in:
[Rossi, R.; Cotela, J.; Lafontaine, Nelson M.; Dadvand, P.; Idelsohn, S. R. Parallel adaptive mesh refinement for incompressible flow problems. Computers and Fluids, 2012]
which has the advantage of being parallel and easy to implement