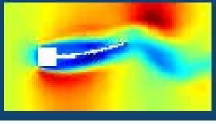


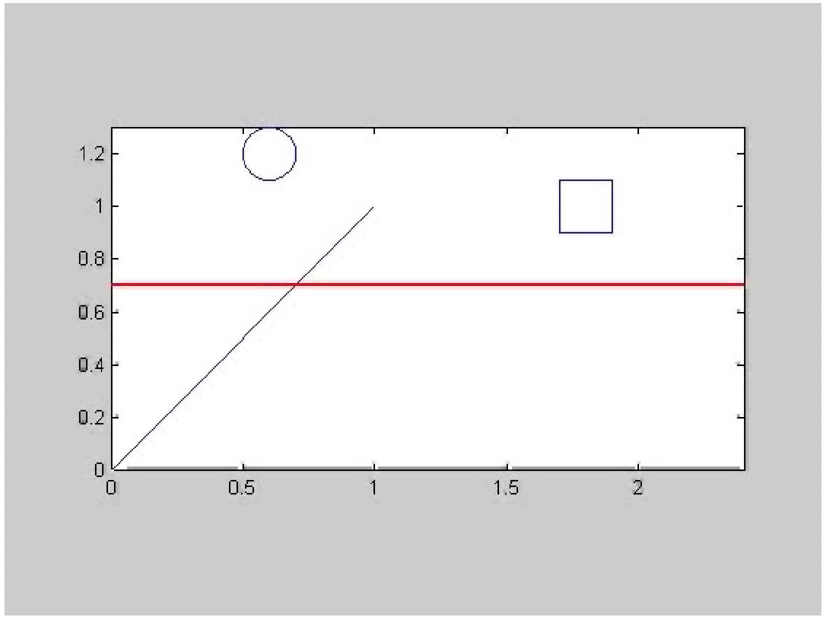
FIXED MESH METHODS IN COMPUTATIONAL MECHANICS

INTRODUCTION

We have seen that ALE formulations are a possibility for dealing with domain movement.

However, after a certain degree of mesh deformation, remeshing is required.

If the mesh generation is carried out by an external program, remeshing every few time steps can be cumbersome.



INTRODUCTION

A possibility to deal with this issue is the use of **Fixed-Mesh methods**

Fixed-Mesh methods are a family of methods where the boundary of the Computational Domain does not coincide with the boundary of the mesh.

This allows to completely decouple the movement of the physical domain from the movement of the mesh (which is fixed in space).

There is no mesh distortion, but the issue of the imposition of Dirichlet boundary conditions appears.

In this section we are going to deal with the Poisson problem in the immersed domain:

$$\begin{aligned} -k\Delta u &= f \qquad && \text{in } \Omega_{\text{in}} \cup \Omega_{\Gamma_{in}} \\ u &= \bar{u}_\Gamma && \text{in } \Gamma \end{aligned}$$

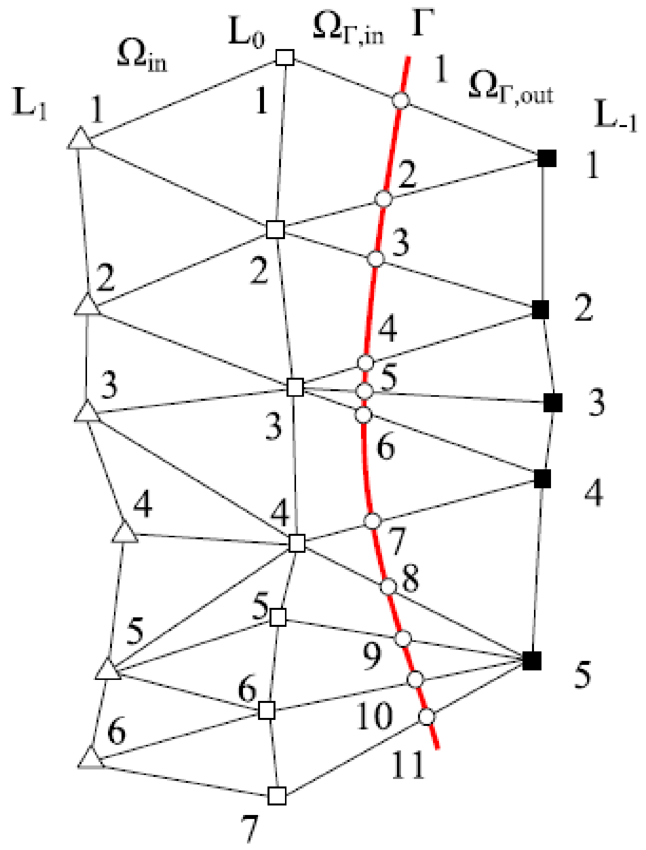


Figure 2.1: Setting

IMPOSITION OF BOUNDARY CONDITIONS

Introduction

One of the key issues is the imposition of boundary conditions. Several families of methods, amongst them:

- Straightforward approach
- Locally remeshing
- Penalty method
- Nitsche's method
- Lagrange multipliers
- Using external degrees of freedom
- Discontinuous Galerkin method

We are going to discuss these methods, their advantages and drawbacks.

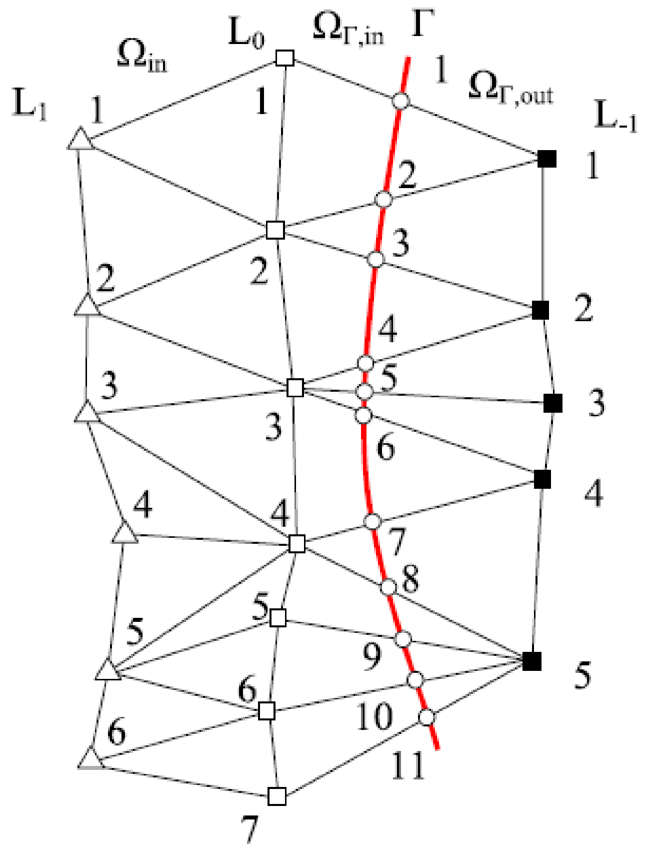


Figure 2.1: Setting

IMPOSITION OF BOUNDARY CONDITIONS

Straight-forward approach

The first approach would be to impose the boundary conditions in those nodes which are closest to the boundary.

For instance, if we are solving the Poisson problem, we solve:

$$\begin{aligned} -k\Delta \mathrm{u} &= \mathrm{f} && \text{in } \Omega_{\mathrm{in}} \cup \Omega_{\Gamma_{in}} \cup \Omega_{\Gamma_{out}} \\ u &= \bar{u}_\Gamma && \text{in } L_{-1} \end{aligned}$$

In this case Dirichlet boundary conditions would be applied as usual, that is by eliminating the rows and columns associated to nodes in L_{-1}.

This approach introduces an error of $\mathcal{O}(h)$!

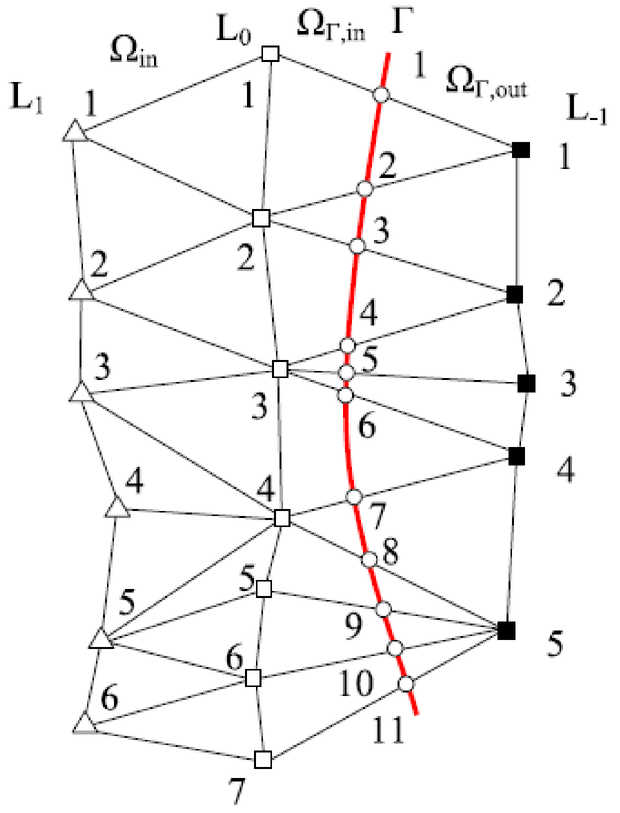
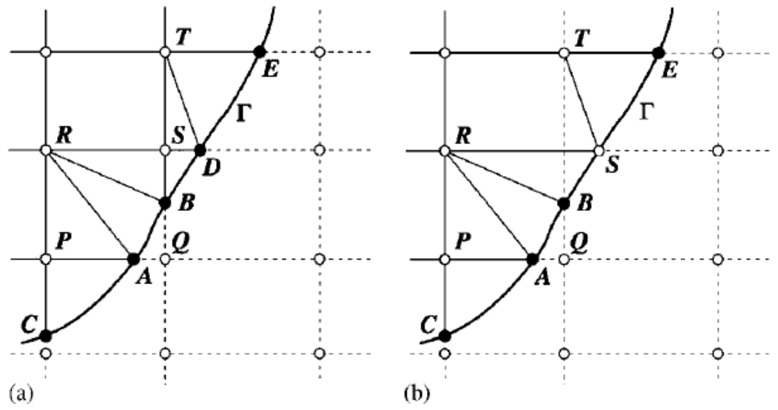


Figure 2.1: Setting

IMPOSITION OF BOUNDARY CONDITIONS

Imposition of Boundary conditions through local remeshing.

Another possibility is to locally modify the mesh in order to make it boundary fitting.



- The main drawback related to this approach is that new nodes and elements need to be built at each time step.
- The local remeshing can be algorithmically complicated to implement in 3D
- The graph and sparsticity of the matrix has to be modified at each time step (memcopy, can be slow).

IMPOSITION OF BOUNDARY CONDITIONS

Penalty method

In the penalty method we work with a penalty term which is added to the variational formulation.

Let $V_h \in H^1(\Omega)$. Let us consider the finite element variational problem: find $u_h \in V_h$ such that:

$$B(v_h, v_h) = k(v_h, u_h) - k < v_h, \boldsymbol{n} \cdot \nabla u >_\Gamma = < v_h, f > \quad \forall v_h \in V_h$$

Remark: Test functions v_h do no longer vanish on Γ*!*

The penalty method adds a term which penalizes the difference between the unknown and the prescribed boundary condition at the interface:

$$B(v_h, v_h) + \alpha < v_h, u_h >_\Gamma = < v_h, f > + \alpha < v_h, \bar{u}_\Gamma >_\Gamma$$

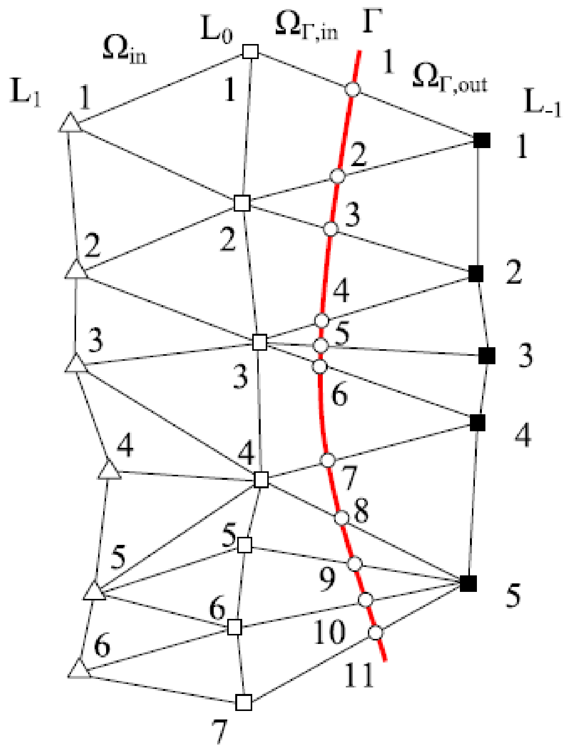


Figure 2.1: Setting

IMPOSITION OF BOUNDARY CONDITIONS

Penalty method

Stability. In order to ensure the stability of the numerical method, the bilinear form needs to be stable.
Let us define the norm:

$$\left|\|u_h\|\right|^2 = k\|\nabla u_h\|_{L^2}{}^2 + \frac{k}{h}\|u_h\|^2_{L^2(\Gamma)}$$

This norm gives us control over $\|u\|_{L^2}$ thanks to Poincaré inequality:

$$\frac{k}{h^2}\|u_h\|_{L^2} \leq C\left(k\|\nabla u_h\|_{L^2}{}^2 + \frac{k}{h}\|u_h\|^2_{L^2(\Gamma)}\right)$$

Stability will be guaranteed if we can assure that:

$$B_{penalty}(u_h, u_h) \geq C\left|\|u_h\|\right|^2$$

IMPOSITION OF BOUNDARY CONDITIONS

Penalty method

$$B_{penalty}(u_h, u_h) = k\|\nabla u_h\|_{L^2}{}^2 - k < u_h, \boldsymbol{n}\cdot\nabla u_h >_\Gamma + \alpha\|u_h\|^2_{L^2(\Gamma)}$$
$$\geq k\|\nabla u_h\|_{L^2}{}^2 - k\|u_h\|_{L^2(\Gamma)}\|\nabla u_h\|_{L^2(\Gamma)} + \alpha\|u_h\|^2_{L^2(\Gamma)}$$

Applying Young's inequality with ϵ/h:

$$B_{penalty}(u_h, u_h) \geq k\|\nabla u_h\|_{L^2}{}^2 - \frac{\epsilon}{2h}k\|u_h\|^2{}_{L^2(\Gamma)} - \frac{h}{2\epsilon}\|\nabla u_h\|_{L^2(\Gamma)} + \alpha\|u_h\|^2_{L^2(\Gamma)}$$
$$\geq k\|\nabla u_h\|_{L^2}{}^2\left(1 - \frac{C}{2\epsilon}\right) + (\alpha - \frac{\epsilon}{2h}k)\|u_h\|^2{}_{L^2(\Gamma)}$$

Stability is assured if:

$$1 - \frac{C}{2\epsilon} > 0$$
$$\alpha - \frac{\epsilon}{2h}k > 0$$

Then we need:

$$\alpha > \frac{C}{h}k$$

C depends on the geometry of the mesh. It can be large. Different expression for different problems.

IMPOSITION OF BOUNDARY CONDITIONS

Nitsche's method

The penalty method is not symmetric. Nitsche's method is a variant of the penalty method which yields a symmetric bilinear form for symmetric problems.

Plus, it takes into account the scaling of the penalty parameter with the mesh size so that the stabilization parameter is mesh size (but not shape) independent.

The associated bilinear form is:

$$B(v_h, v_h) + \alpha \frac{k}{h} < v_h, u_h >_\Gamma - k < \boldsymbol{n} \cdot \nabla v_h, u_h >_\Gamma =$$

$$< v_h, f > + \alpha \frac{k}{h} < v_h, \bar{u}_\Gamma >_\Gamma - k < \boldsymbol{n} \cdot \nabla v_h, \bar{u}_\Gamma >_\Gamma \qquad \forall v_h \in V_h$$

The green terms are enforcing the Dirichlet boundary conditions, tested against $\boldsymbol{n} \cdot \nabla v_h$. The term in the LHS is the symmetric counterpart of the fluxes in the Galerkin form of the problem.

In order for the method to be stable we require: $\alpha > C$.

It is an improvement, but we still need to estimate the value for C (geometry dependent). Plus, stability estimate is different for each problem of interest.

IMPOSITION OF BOUNDARY CONDITIONS

Lagrange multipliers

Another possibility for enforcing the Dirichlet boundary conditions is the use of Lagrange multipliers.

Let $V_h \subset H^1(\Omega), W_h' \subset H^{-1/2}(\Gamma)$. The Lagrange multipliers bilinear form is: find $u_h \in V_h$ and $\lambda_h \in W'$ such that:

$$k(\nabla v_h, \nabla u_h) - < v_h, \lambda_h >_\Gamma = < v_h, f > \qquad \forall v_h \in V_h$$
$$< \gamma_h, u_h - \bar{u}_\Gamma >_\Gamma = 0 \qquad \forall \gamma_h \in W_h'$$

$$B_{LM}([v_h, \gamma_h], [u_h, \lambda_h]) = k(\nabla v_h, \nabla u_h) - < v_h, \lambda_h >_\Gamma + < u_h, \gamma_h >_\Gamma$$

If we test the coercivity of the method we find:

$$B_{LM}([u_h, \lambda_h], [u_h, \lambda_h]) = k\|\nabla u_h\|^2$$

We don't have any control over $\|u_h\|_{L^2(\Gamma)}$ or $\|\lambda_h\|_{W_h'}$. We need an inf-sup condition:

$$\inf_{\lambda_h \in W_h'} \sup_{u_h \in V_h} \frac{< u_h, \lambda_h >_\Gamma}{\|u_h\|_{L^2(\Gamma)} \|\lambda_h\|_{W_h'}} \geq C > 0$$

IMPOSITION OF BOUNDARY CONDITIONS

Lagrange multipliers

- Lagrange multipliers can be used effectively to impose Dirichlet boundary conditions with optimal convergence order.
- There is no penalty parameter, no problem dependent parameter.

On the other hand:

- Additional degrees of freedom need to be introduced to the system of equations.
- The resulting problem is a saddle point problem, which is known to have ill-conditioning issues for its solution with iterative solvers.

It is also possible to devise a stabilization mechanism for the Lagrange multipliers, which allows us to get rid from the inf-sup condition (Barbosa-Hughes 1990):

$$B_{LMS}([v_h,\gamma_h],[u_h,\lambda_h]) = B_{LM}([v_h,\gamma_h],[u_h,\lambda_h]) + B_S([v_h,\gamma_h],[u_h,\lambda_h])$$

$$B_S([v_h,\gamma_h],[u_h,\lambda_h]) = -\delta h^2 < n\cdot\nabla u+\lambda_h, \mathrm{n}\cdot\nabla v_h+\gamma_h >_\Gamma +\delta_2 h^2 < v_h, u_h-\bar{u} >_\Gamma$$

These stabilization terms are similar to Nitsche's method.

IMPOSITION OF BOUNDARY CONDITIONS

A method for strongly enforcing Dirichlet boundary conditions in immersed boundary methods.

The idea is similar to what is done in boundary fitting methods (equation elimination). We start by re-writting Nitsche's method:

Consider the splitting $V_h = V_{h,0} \oplus V_{h,\Gamma}$ with $V_{h,0}$ the subspace of functions vanishing at the nodes outside Ω_{in} , including its boundary, and $V_{h,\Gamma}$ the complement

$$\boxed{B(u_{h,0}, v_{h,0}) - k\langle \partial_n u_{h,0}, v_{h,0}\rangle_\Gamma + B(u_{h,\Gamma}, v_{h,0}) - k\langle \partial_n u_{h,\Gamma}, v_{h,0}\rangle_\Gamma = \langle f, v_{h,0}\rangle_\Omega,}$$

$$B(u_{h,0}, v_{h,\Gamma}) - k\langle \partial_n u_{h,0}, v_{h,\Gamma}\rangle_\Gamma + B(u_{h,\Gamma}, v_{h,\Gamma}) - k\langle \partial_n u_{h,\Gamma}, v_{h,\Gamma}\rangle_\Gamma = \langle f, v_{h,\Gamma}\rangle_\Omega,$$

$$- k\langle \partial_n v_{h,0}, u_{h,0}\rangle_\Gamma - k\langle \partial_n v_{h,0}, u_{h,\Gamma}\rangle_\Gamma = -k\langle \partial_n v_{h,0}, \bar{u}\rangle_\Gamma,$$

$$- k\langle \partial_n v_{h,\Gamma}, u_{h,0}\rangle_\Gamma - k\langle \partial_n v_{h,\Gamma}, u_{h,\Gamma}\rangle_\Gamma = -k\langle \partial_n v_{h,\Gamma}, \bar{u}\rangle_\Gamma,$$

$$\frac{\alpha k^*}{h}\langle u_{h,0}, v_{h,0}\rangle_\Gamma + \frac{\alpha k^*}{h}\langle u_{h,\Gamma}, v_{h,0}\rangle_\Gamma = \frac{\alpha k^*}{h}\langle \bar{u}, v_{h,0}\rangle_\Gamma,$$

$$\boxed{\frac{\alpha k^*}{h}\langle u_{h,0}, v_{h,\Gamma}\rangle_\Gamma + \frac{\alpha k^*}{h}\langle u_{h,\Gamma}, v_{h,\Gamma}\rangle_\Gamma = \frac{\alpha k^*}{h}\langle \bar{u}, v_{h,\Gamma}\rangle_\Gamma.}$$

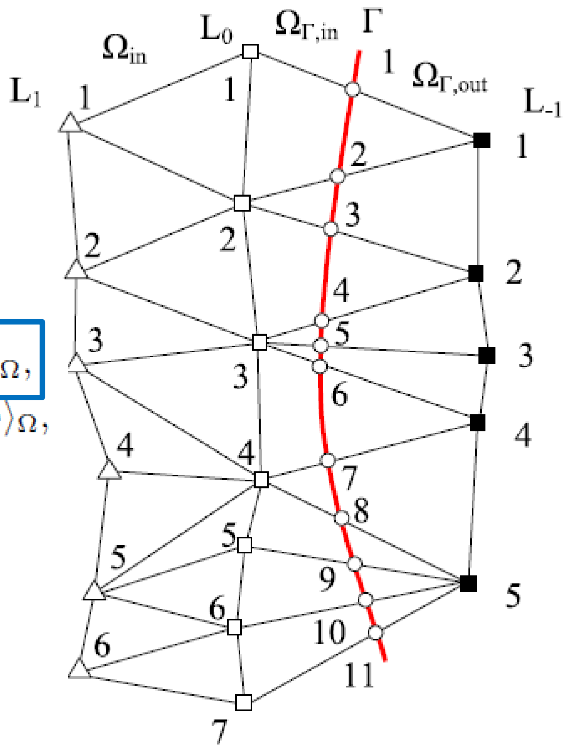


Figure 2.1: Setting

We are using the degrees of freedom associated to external nodes in order to minimize $||u_h - \bar{u}||_{L^2(\Gamma)}$.

IMPOSITION OF BOUNDARY CONDITIONS

A method for strongly enforcing Dirichlet boundary conditions in immersed boundary methods.

Find $u_{h,0} \in V_{h,0}$ and $u_{h,\Gamma} \in V_{h,\Gamma}$ such that

$$B(u_{h,0}, v_{h,0}) + B(u_{h,\Gamma}, v_{h,0}) - k\langle \partial_n u_{h,0}, v_{h,0}\rangle_\Gamma - k\langle \partial_n u_{h,\Gamma}, v_{h,0}\rangle_\Gamma = \langle f, v_{h,0}\rangle_\Omega$$

$$\frac{\alpha k^*}{h}\langle u_{h,0}, v_{h,\Gamma}\rangle_\Gamma + \frac{\alpha k^*}{h}\langle u_{h,\Gamma}, v_{h,\Gamma}\rangle_\Gamma = \frac{\alpha k^*}{h}\langle \bar{u}, v_{h,\Gamma}\rangle_\Gamma$$

for all $v_{h,0} \in V_{h,0}$ and $v_{h,\Gamma} \in V_{h,\Gamma}$.

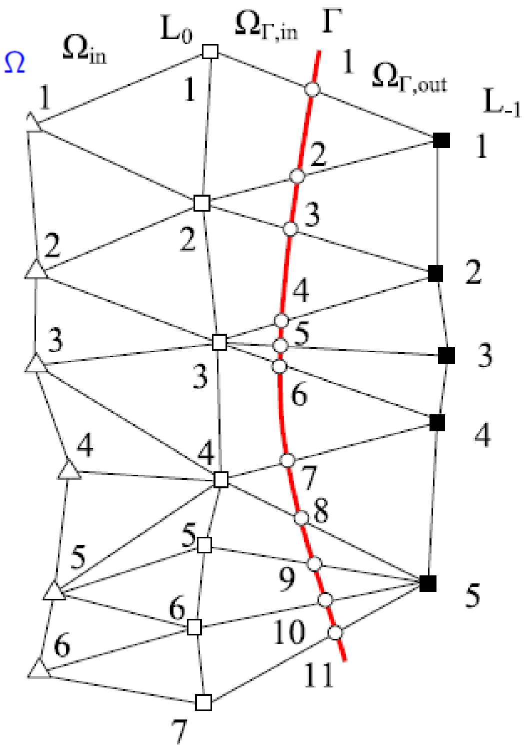


Figure 2.1: Setting

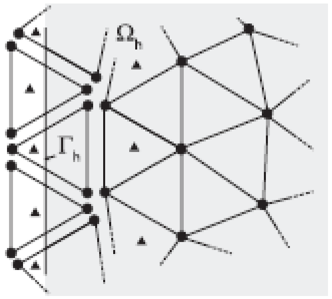
Properties:

- When Γ coincides with $\partial\Omega_h$ the boundary conditions is imposed exactly (provided $\bar{u}$ is a finite element function
- There are no parameters to be tuned ($\frac{\alpha k*}{h}$ can be dropped).
- The method is non-symmetric even if B is symmetric.
- There are no additional degrees of freedom
- The method can be shown to be stable.
- Optimal order of accuracy is obtained (quadratic convergence for linear elements).

IMPOSITION OF BOUNDARY CONDITIONS

Discontinuous-Galerkin based imposition of boundary conditions (Lew-Buscaglia 2007)

Discontinuous-Galerkin interpolation spaces are not continuous across the element interfaces.
This results in a richer interpolation space, but it also requires the modeling of the interelement fluxes.



From Lew and Buscaglia 2007

We use continuous Galerkin everywhere except for those elements which are cut by the exterior boundary Γ.

$$B(v_h, v_h) + [\![v_h, \boldsymbol{n} \cdot \nabla u_h]\!] + \alpha[\![v_h, u_h]\!] =< v_h, f >$$

We use the equations associated to exterior nodes to enforce that the interpolated values at the interface are exactly $\bar{u}$.
However, additional degrees of freedom are required.

FAMILIES OF FIXED-MESH METHODS IN COMPUTATIONAL MECHANICS

Introduction.

The definition of the families is many times related to the method used for the imposition of boundary conditions.

Several families of methods exist:

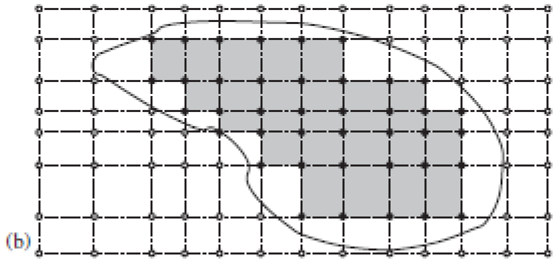
- Immersed boundary method.
- Fictitious domain method.
- Physical Domain methods:
    - Extrapolation method.
    - Fixed-Mesh ALE.
- Chimera type strategies.

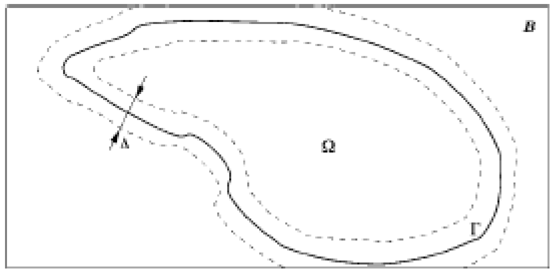
Again, all of them have advantages and drawbacks.

FAMILIES OF FIXED-MESH METHODS IN COMPUTATIONAL MECHANICS

The immersed boundary method

In the immersed boundary method, we solve the equations of interest over the domain covered by the mesh (not the physical domain).





The effect of the immersed solid body on the fluid is taken into account through a penalty force.

$$\boldsymbol{f} = \sum_{i=1}^{P} k(\boldsymbol{u} - \bar{\boldsymbol{u}})\delta(|\boldsymbol{x} - \boldsymbol{x}_{\Gamma,i}|),$$

δ represents the Dirac-delta function. Summation is over all the nodes at the interface.

δ is smoothed to a more or less sharp function over the whole computational domain.

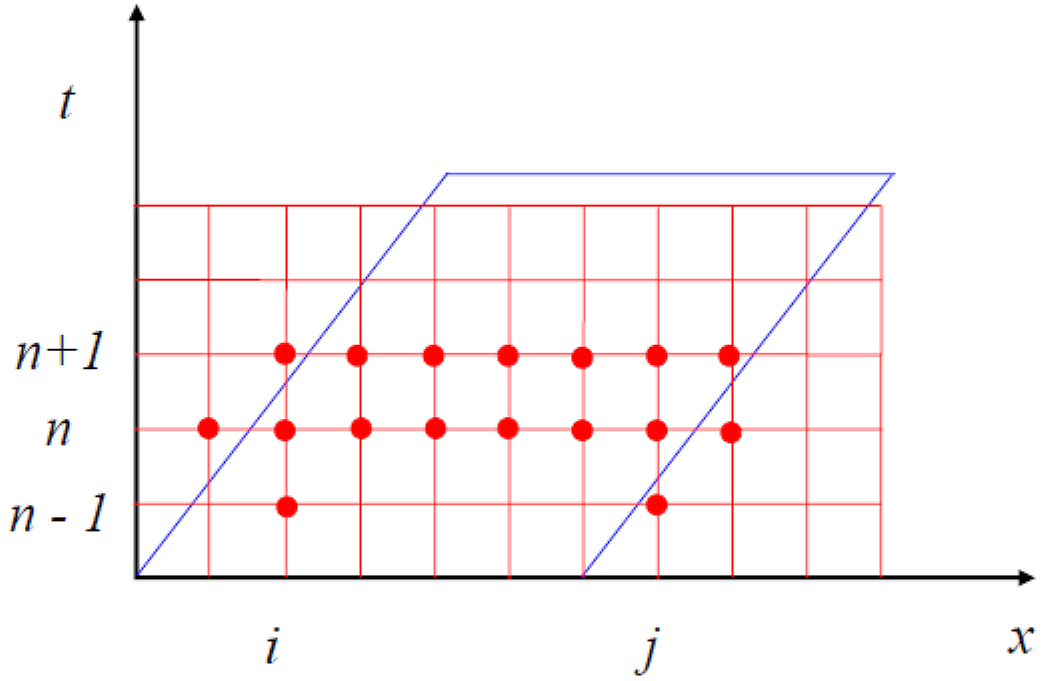
FAMILIES OF FIXED-MESH METHODS IN COMPUTATIONAL MECHANICS

The immersed boundary method

- The part of the domain occupied by the solid body is also solved when computing for the fluid.
- This is inaccurate, because we are taking into account the effect of a volume of fluid which is not there.
- On the other hand, this can help with the added mass effect if a partitioned fluid-structure interaction approach is taken.

FAMILIES OF FIXED-MESH METHODS IN COMPUTATIONAL MECHANICS

The issue of newly created nodes



In the immersed boundary method, newly created nodes are treated by using the velocity in the solid body at the previous time step.

FAMILIES OF FIXED-MESH METHODS IN COMPUTATIONAL MECHANICS

The Fictitious Domain method

The fictitious domain method is very similar to the immersed boundary method.

It also solves the fluid equations over the whole computational domain.

The main differences are the following:

- Boundary conditions are applied by using a Lagrange multiplier technique.
- Velocity values in the newly created nodes are taken from the fluid solution in the fictitious solid domain.

FAMILIES OF FIXED-MESH METHODS IN COMPUTATIONAL MECHANICS

Physical domain methods

In this family of Fixed-Mesh methods, the domain over which we integrate the finite element equations exactly coincides with the physical domain:

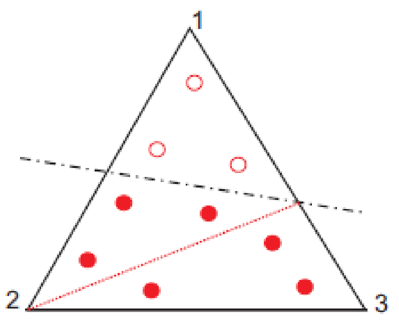


Figure 4.3: Splitting of elements

This introduces the need of subintegration: we introduce new Gauss Points, but we keep the unknowns of the problem.

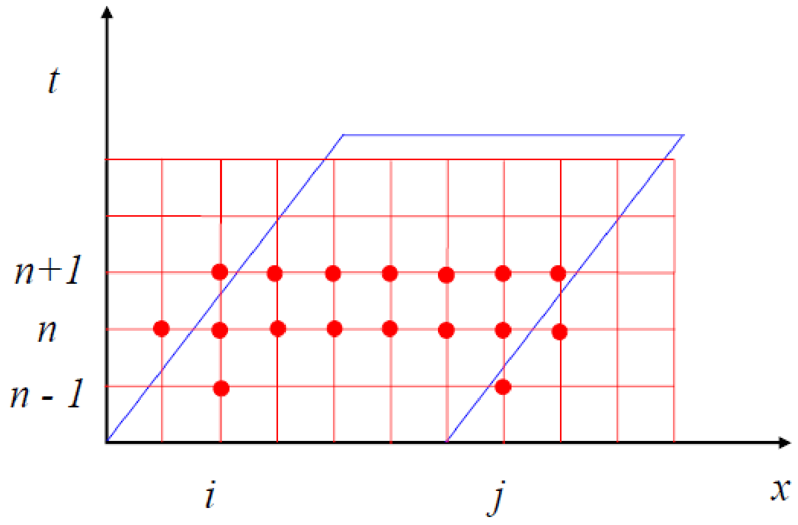
The velocity unknown is not defined outside the physical domain. What to do with newly created nodes?

FAMILIES OF FIXED-MESH METHODS IN COMPUTATIONAL MECHANICS

First approach: extrapolation method.

In the extrapolation method, the value of the velocity at the previous time step in newly created nodes is extrapolated.

If the time step is small the introduced error is also small. But for large time steps, it can lead to the apparition of spurious solutions. (Large gradients at the boundary layer)



FAMILIES OF FIXED-MESH METHODS IN COMPUTATIONAL MECHANICS

Second approach: The Fixed-Mesh ALE method

The Fixed-Mesh ALE method is a hybrid between fixed-mesh and ALE strategies.

Its motivation is due to the issue of newly created nodes.

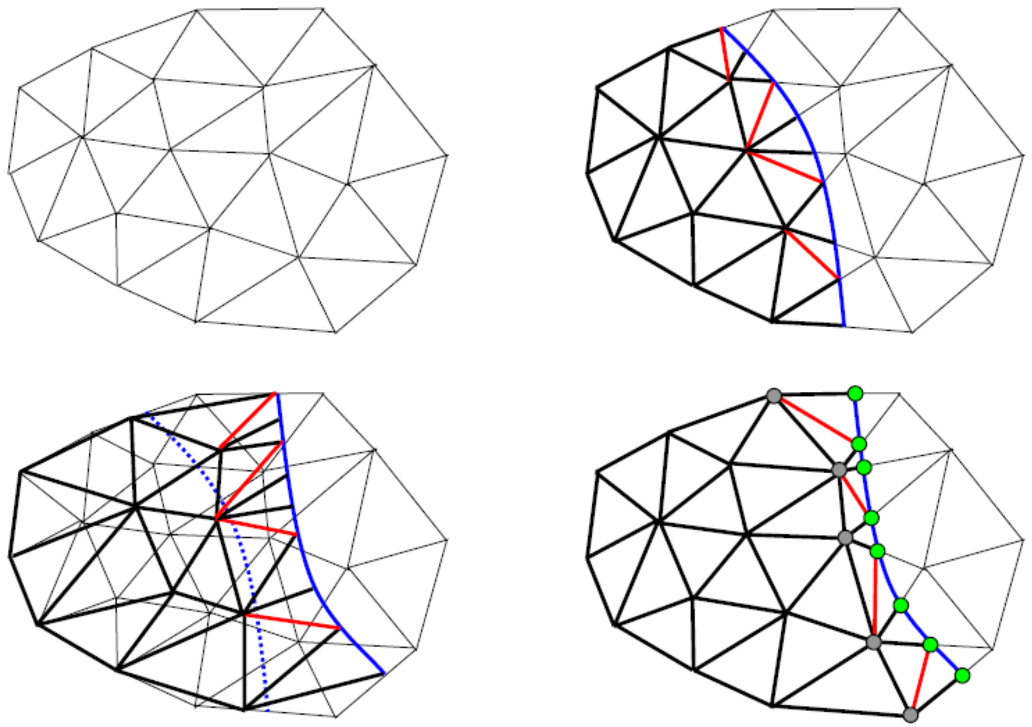
Suppose Ω^0 is meshed with a finite element mesh M^0 and that at time t^n the domain Ω^n is meshed with a finite element mesh M^n. Let $\boldsymbol{u}^n$ be the velocity already computed on Ω^n.

1. Define $\Gamma_{\mathrm{free}}^{n+1}$ by updating a boundary function.
2. Deform virtually the mesh M^n to M_{virt}^{n+1} using the classical ALE concepts and compute the mesh velocity $\boldsymbol{u}_m^{n+1}$.
3. Write down the ALE Navier-Stokes equations on M_{virt}^{n+1}.
4. Split the elements of M^0 cut by $\Gamma_{\mathrm{free}}^{n+1}$ to define a mesh on Ω^{n+1}, M^{n+1}.
5. Project the ALE Navier-Stokes equations from M_{virt}^{n+1} to M^{n+1}.
6. Solve the equations on M^{n+1} to compute $\boldsymbol{u}^{n+1}$ and p^{n+1}.

FAMILIES OF FIXED-MESH METHODS IN COMPUTATIONAL MECHANICS

Second approach: The Fixed-Mesh ALE method

Top left: M^0. Top right: M^n. Bot. left: M_{virt}^{n+1}. Bot. right: M^{n+1}

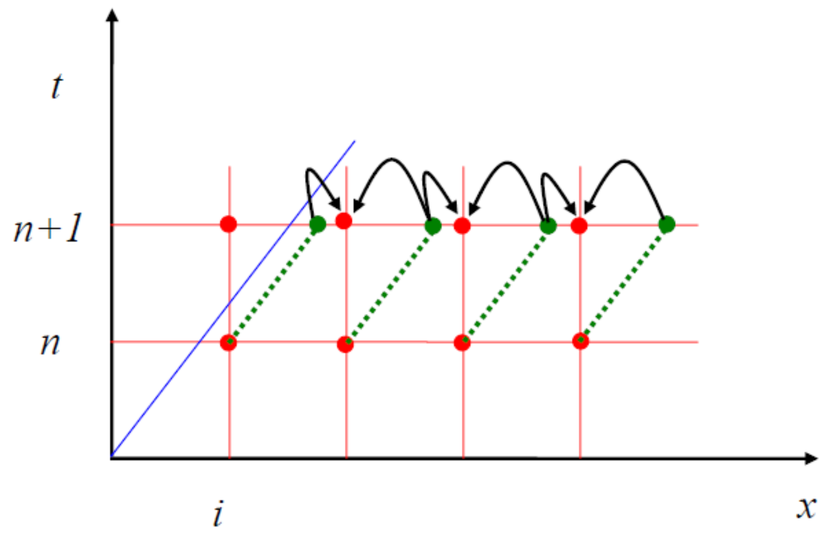


FAMILIES OF FIXED-MESH METHODS IN COMPUTATIONAL MECHANICS

Second approach: The Fixed-Mesh ALE method

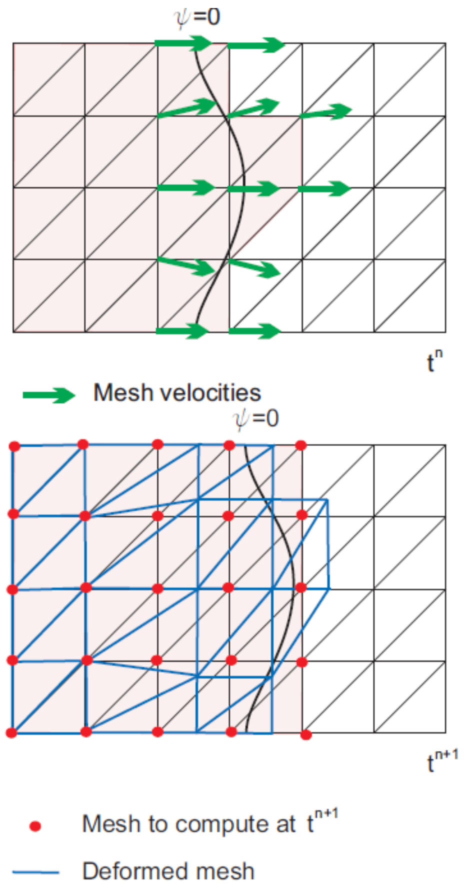
Sketch of the method:

- Red nodes at t^n are moved to green nodes at t^{n+1}
- Values at red nodes at t^{n+1} (possibly of functions at t^n) are interpolated from values at green nodes.



FAMILIES OF FIXED-MESH METHODS IN COMPUTATIONAL MECHANICS

Second approach: The Fixed-Mesh ALE method
Step 1: Mesh velocity



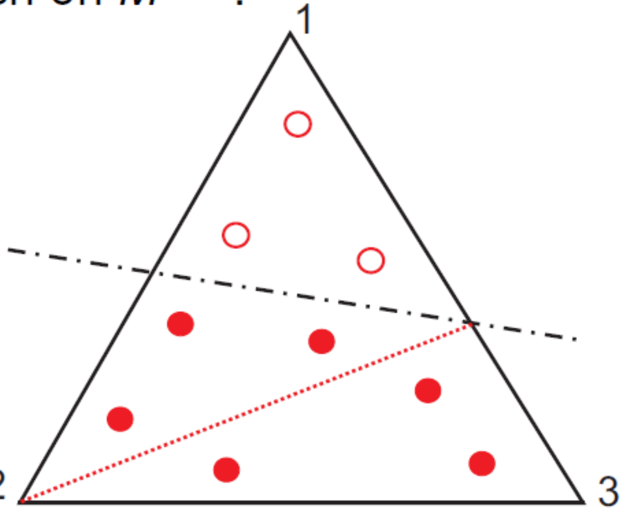
- Updating the boundary defines the deformation of the domain from Ω^n to Ω^{n+1}.
- The mesh velocity on the boundary points can be computed from their position $\boldsymbol{x}^{n+1}$ and $\boldsymbol{x}^n$: $\boldsymbol{u}_m = (\boldsymbol{x}^{n+1} - \boldsymbol{x}^n)/\delta t$.
- On the rest of the nodes, $\boldsymbol{u}_m$ can be computed solving $\Delta \boldsymbol{u}_m = 0$. It is also possible to restrict $\boldsymbol{u}_m \neq \boldsymbol{0}$ to the nodes next to $\Gamma^{n+1}_{\text{free}}$.

FAMILIES OF FIXED-MESH METHODS IN COMPUTATIONAL MECHANICS

Second approach: The Fixed-Mesh ALE method

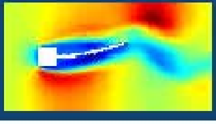
Step 2: Element splitting and approximate boundary conditions

Elements on the background mesh M^0 cut by $\Gamma^{n+1}_{\text{free}}$ are split to define a mesh on M^{n+1}.



Both M^{n+1} and M^{n+1}_{virt} are meshes of the domain Ω^{n+1}. M^{n+1} is a minor modification of the background mesh M^0.

Boundary conditions need to be enforced at the interface: Nitsche, Strong imposition…



FAMILIES OF FIXED-MESH METHODS IN COMPUTATIONAL MECHANICS

Second approach: The Fixed-Mesh ALE method

Step 3: Unknown values projection

The velocity $\boldsymbol{u}^n$ in M_{virt}^{n+1} is known because its nodal values correspond to those of mesh M^n.
Let P^{n+1} be the projection from finite element functions from M_{virt}^{n+1} to M^{n+1}. Denoting again $\boldsymbol{u}^{n+1} \equiv P^{n+1}(\boldsymbol{u}^{n+1})$, the flow equations on M^{n+1} are

$$\rho\left[\frac{1}{\delta t}(\boldsymbol{u}^{n+1} - P^{n+1}(\boldsymbol{u}^n)) + (\boldsymbol{u}^{n+1} - P^{n+1}(\boldsymbol{u}_m)) \cdot \nabla \boldsymbol{u}^{n+1}\right]$$
$$- \nabla \cdot [2\mu \nabla^S(\boldsymbol{u}^{n+1})] + \nabla p^{n+1} = \boldsymbol{f}$$
$$\nabla \cdot \boldsymbol{u}^{n+1} = 0$$

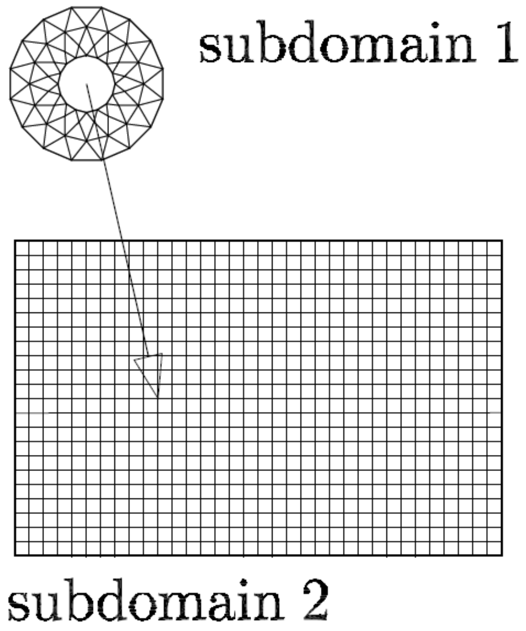
in Ω^{n+1} with boundary conditions on $\Gamma_{\mathrm{free}}^{n+1}$.
The only difficulty is to compute $P^{n+1}(\boldsymbol{u}^n)$, wich can be done by identifying nodes from M^{n+1} on M_{virt}^{n+1} and using interpolation.

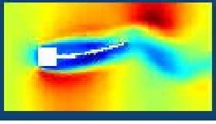
FAMILIES OF FIXED-MESH METHODS IN COMPUTATIONAL MECHANICS

Chimera type methods

The main features of Chimera methods are the following:

- Independent subdomains are defined around each object and for the fixed domain in which the objects are moving.
- A domain decomposition technique is used to couple at each time step the solution obtained on the subdomains.
- The new positions and subdomain linear and angular accelerations and velocities are computed by integrating the equations of motion for the solids





FAMILIES OF FIXED-MESH METHODS IN COMPUTATIONAL MECHANICS

Chimera type methods

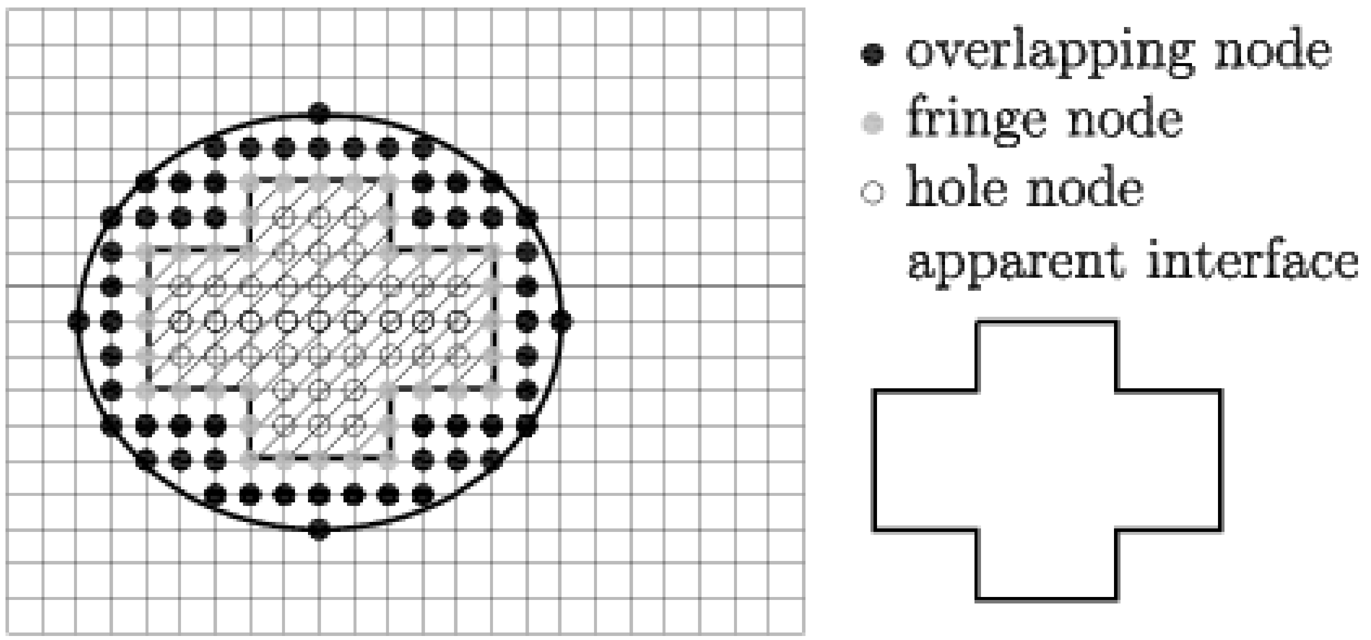
Some terminology:

- Mesh used to discretize the fixed domain Ω: background mesh
- Mesh used to discretize the moving domain Ω_1 around the body: patch mesh.
- Set of the two overset grids: composite grid.
- Remove some elements of the background located inside the patch in order to define an apparent interface; this task is called hole cutting.
- Nodes forming the apparent interface: fringe nodes.
- Nodes interior to the hole: hole nodes.
- Set of hole nodes and fringe nodes: interpolation nodes.
- Nodes between the apparent interface and the outer boundary of the patch mesh: overlapping nodes.

FAMILIES OF FIXED-MESH METHODS IN COMPUTATIONAL MECHANICS

Chimera type methods

Some terminology:



FAMILIES OF FIXED-MESH METHODS IN COMPUTATIONAL MECHANICS

Chimera type methods

Transmission conditions

- $\partial\Omega_1$ will be smooth. Neumann or Robin conditions can be prescribed.
- The apparent interface, in practice part of $\partial\Omega_2$ will not be smooth. Only Dirichlet conditions can be prescribed.

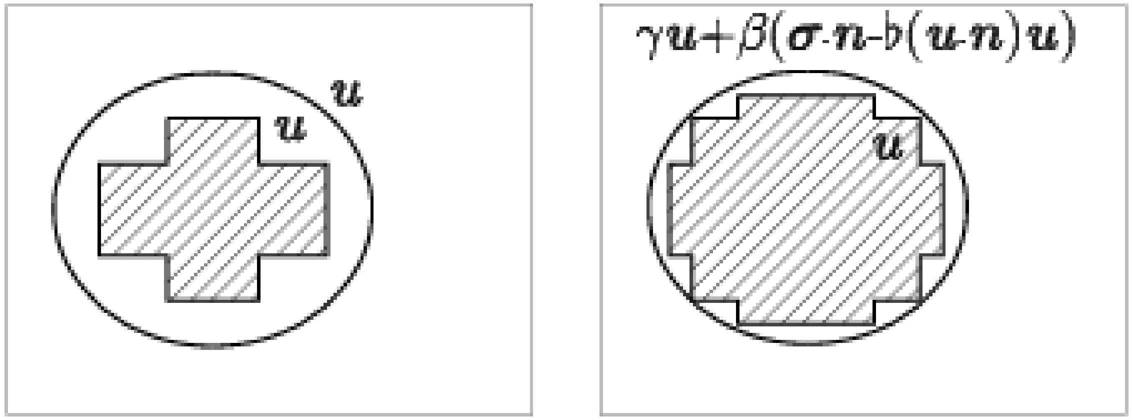

$\boldsymbol{u}$
$\boldsymbol{u}$
$\gamma\boldsymbol{u}+\beta(\boldsymbol{\sigma}\cdot\boldsymbol{n}-\flat(\boldsymbol{u}\cdot\boldsymbol{n})\boldsymbol{u})$
$\boldsymbol{u}$

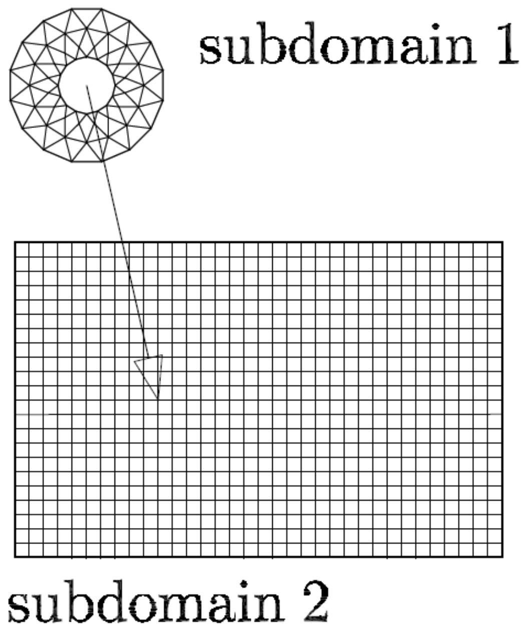
Figure 4: *Chimera method. Hole and variables transmitted. (Left) Chimera/Dirichlet. (Right) Chimera/Neumann ($\gamma = 0$) or Chimera/Robin ($\gamma \neq 0$).*

FAMILIES OF FIXED-MESH METHODS IN COMPUTATIONAL MECHANICS

Chimera type methods

Since there is overlapping, several transmission conditions can be applied:

- Dirichlet – Dirichlet conditions in the boundaries.
  - Convergence of the iteration-by-subdomain dependent on the overlapping region.
- Dirichlet – Neumann conditions with overlapping
  - Better convergence of the iteration by subdomain.



But also:

- Dirichlet – Robin
- Robin – Robin

It is obviously possible to solve both meshes using a monolithic scheme.
The inconvenient is that at each time step the connectivities of the mesh change (also the graph of the global matrix).

The main advantage of the method is that it allows to refine in the boundary layer.