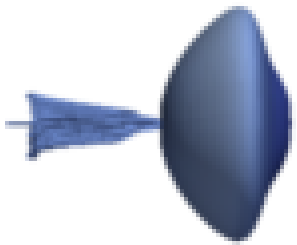
Boundary conditions in LifeV

State of the art and main issues

Guido Iori

LifeV Meeting 2014 - Politecnico di Milano

7th January, 2014



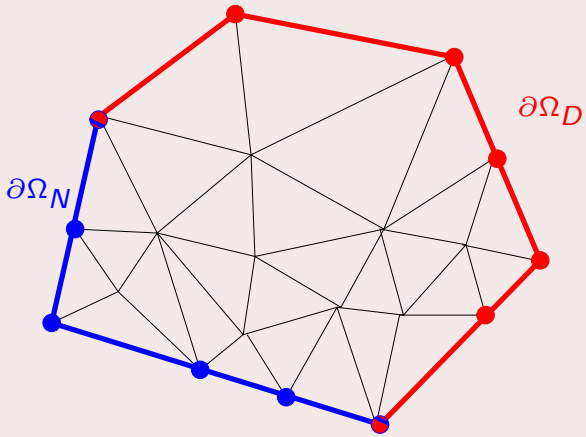
Outline

Imposing and implementing BC... hard stuff!

Modelling aspects (not our problem, at least today!)

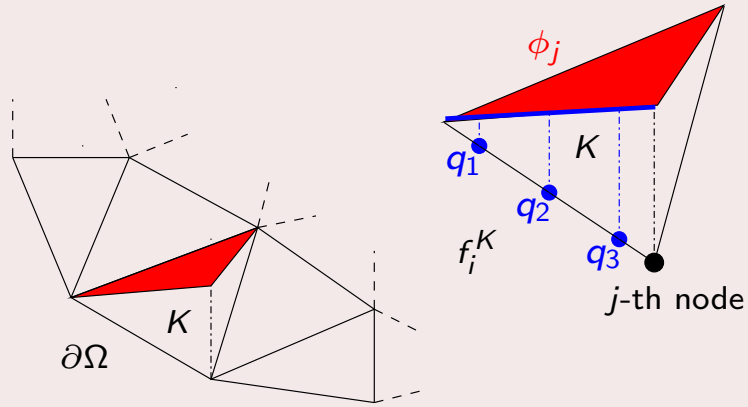
Well position of the differential problem, condition of compatibility...

Geometrical informations



- approximation of boundary $\partial\Omega$ by a polygonal line,
- definition of boundaries $\partial\Omega_N, \partial\Omega_D, \ldots$ w.r.t. the imposed BC condition,
- information about facets ($n-1$ geometrical entities)

FEM informations



- identification of (boundary) degrees of freedom by facet
- evaluation of ϕ_j on the quadrature nodes q_i on facet f_i^K for integration (no more elemental operators)

Essential conditions

Dirichlet b.c.

$$u = f \quad \text{on} \quad \partial\Omega_D \tag{1}$$

To impose (1) on the i-th (global) DOF, we need to:

- evaluate $f(x_i)$ where x_i is the point in the domain Ω associated to the i-th DOF,
- put $k \geq 1$ in the cell (i, i) of the global matrix and 0 in the other cells of the i-th row,
- put $kf(x_i)$ in the i-th row of the right-hand side global vector.

$$\begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ 0 & 0 & 1 & 0 \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ \cdots \\ u_m \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ f(x_i) \\ b_m \end{pmatrix}$$

This is implemented in `LifeV`, but we remark that is not the only way to impose Dirichlet boundary condition.

Natural conditions

Neumann b.c.

$$\nabla \mathbf{u} \cdot \mathbf{n} = g \quad \text{on} \quad \partial\Omega_N \tag{2}$$

To impose (2), we need to loop on all Neumann boundary facets and:

- identify the boundary DOFs of the j-th facet
- compute $\int_{\partial\Omega_N} g\phi_i d\gamma$ for all ϕ_i having support on the j-th facet ,
- add $\int_{\partial\Omega_N} g\phi_i d\gamma$ to the i-th row of the right-hand side global vector.

$$\begin{pmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ \cdots \\ u_m \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_i + \int_{\partial\Omega_N} g\phi_i d\gamma \\ b_m \end{pmatrix}$$

Data structures

In `LifeV`, data structures operating to impose the boundary conditions are denoted by " BC " prefix

- `BCFunctionBase` and children, defining functors that specify the BC
- `BCIdentifierBase` and children, storing information about the DOFs involved in BC. They may differ according to BC type
- `BCBase`
- `BCHandler`
- `bcManage`

How to extricate in this BC jungle???

Functions

`BCFunctionBase` (parent class): it's a functor with call operator

```
Real f(const Real& t, const Real& x, const Real& y, const Real& z, const ID& comp)
```

and its *children*

- `BCDataInterpolator`: implements Radial Basis Function (RBF) interpolation of pointwise scalar or vectorial functions defined on a set of scattered interpolation points
- `BCFunctionDirectional`: used for prescribing essential boundary conditions along a direction, a versor function is needed to define the direction (for instance, normal or tangential versor to the boundary layer)
- `BCFunctionRobin`: functions used for prescribing Robin boundary conditions, additional coefficient functions are needed. For instance, to prescribe the following condition,

$$a\mathbf{u} + b\nabla_{\mathbf{n}}\mathbf{u} = g \quad \text{on } \partial\Omega$$

we need to provide coefficients a and b.

DOFs for BC

`BCIdentifierBase` (parent class): it contains an ID (DOF for essential BC, facet ID for natural BC),
and its *children*

- `BCIdentifierEssential`: coordinates x, y, z
- `BCIdentifierNatural`: map of DOFs of the facet

These classes are used to store the IDs of DOFs involved by a BC (needed to assemble the contribution in the correct cells of global matrix/vector) and the coordinates for Dirichlet BC (where the evaluation of a function on the boundary is needed).

Storing informations about a BC

`BCBase`, it's a class containing all the information about a boundary condition:

- name
- type (Natural, Robin, Flux, Resistance, Essential, EssentialEdges, EssentialVertices)
- mode (Scalar, Full, Component, Normal, Tangential, Directional)
- components (x, y, z)
- vector of coefficients or pointer to a function (`BCFunctionBase` or children)
- list of identifiers (`BCIdentifierBase` or children)

BCHandler

BCHandler is a container of BCBase objects. It instantiates the needed BCBase objects and provide them with the necessary data.

As an example, if a BCBase object is related to an Essential boundary condition, then the global DOF IDs associated with that essential boundary condition will be passed to the object.

```
BCFunctionBase bcFuncEss(solOnBoundary), bcFuncNat(zeroFunction);
BCHandler bcHandler;

bcHandler.addBC( "inlet", 10, Essential, Full, bcFuncEss, 3 );
bcHandler.addBC( "outlet", 20, Natural, Full, bcFuncNat, 3 );
// 10, 20, 30 refers to the markers (boundary flags)
// 3 is the number of dimensions of the solution field

bcHandler.bcUpdate(mesh, boundaryFe, dof);
```

bcManage

It's a free function that, using the informations stored in a `BCHandler` object, implements the modifications to the global matrix and global vector to impose the boundary conditions.

Declaration and definition are contained in `BCManage.hpp` and `BCManage.cpp`.

```
template<typename MatrixType, typename VectorType,
                    typename MeshType, typename DataType>

void bcManage (MatrixType &matrix, VectorType &rightHandSide,
               MeshType const &mesh,
               DOF const &dof,
               BCHandler const &bcHandler,
               CurrentFEManifold &currentBdFE,
               DataType const &diagonalizeCoef, //diagonal entry for Dirichlet BC
               DataType const &time=0)
```

A complete example

```
BCHandler bchandler;
BCFunctionBase BCu( exactSolution );
bchandler.addBC("Dirichlet",1,Essential,Full,BCu,1);
for (UInt i(2); i<=6; ++i)
{
     bchandler.addBC("Dirichlet",i,Essential,Full,BCu,1);
}

bchandler.bcUpdate(*uFESpace->mesh(),uFESpace->feBd(),uFESpace->dof());

vector_Type rhsBC(rhs,Unique);
bcManage(*systemMatrix,rhsBC,*uFESpace->mesh(),uFESpace->dof(),
             bchandler,uFESpace->feBd(),1.0,0.0);

rhs = rhsBC;
```

For further information, read the DoxyGen!

Useful remarks

- It is possible then the same DOF is shared by different boundary conditions. In the case of essential boundary conditions, *by default*, the condition with largest flag will be prescribed on the shared DOF. In particular, if two Essential boundary conditions share the same DOF, it will be prescribed the condition with the largest flag [1]. This behavior is due to the fact that the largest boundary condition is the last to be prescribed.
- Facet DOF IDs are stored in the `DOF` class member `M_localToGlobalByBdFacet`. Then, these information are copied in the local maps of `BCIdentifierNatural` objects.

[1] This behaviour can be changed by the user by modifying the policy (see `MarkerIDStandardPolicy` class).

Todo list for BC

- cleaning of useless classes (`BCFunctionUDepRobin`)
- computation of boundary contributions by integration of the trace of nD basis function on the $(n-1)D$ geometrical entities (facets). The current implementation integrate the basis function of the facet, for $\mathbb{RT}0$ elements this is not correct.

Topics to be discussed

- why storing all the geometrical and DOF informations if we can use `FESpace` class?
- allow to choose different kind of BC functions (time, coordinates and component cannot be sufficient to evaluate a function, for instance XFEM needs another parameter)
- why `bcManage` is a free function?

Any other issues or weird behaviours?