FC-VFEM$\mathbb{P}_1$ Python package, User's Guide [1]

François Cuvelier[2]

2017/06/23

[1]Compiled with Python 3.6.0

[2]Université Paris 13, Sorbonne Paris Cité, LAGA, CNRS UMR 7539, 99 Avenue J-B Clément, F-93430 Villetaneuse, France, cuvelier@math.univ-paris13.fr.

This work was partially supported by ANR Dedales.

Abstract

FC-VFEM$\mathbb{P}_1$ is an **experimental** object-oriented Python package dedicated to solve scalar or vector boundary value problems (BVP) by $\mathbb{P}^1$-Lagrange finite element method in any space dimension. It uses the FC-SIMESH package [?] and the `siMesh` class which allows to use simplices meshes generated from gmsh (in dimension 2 or 3) or an hypercube triangulation (in any dimension).

The two FC-SIMESH add-ons FC-SIMESH-MATPLOTLIB [?] and FC-SIMESH-MAYAVI [?] allows a great flexibility in graphical representations of the meshes and datas on the meshes by using respectively the MATPLOTLIB and the MAYAVI packages.

This package also contains the techniques of vectorization presented in [?] and extended in [?] and allows good performances when using $\mathbb{P}^1$-Lagrange finite elements method.

Contents

1 Generic Boundary Value Problems **3**
1.1 Scalar boundary value problem . . . 3
1.2 Vector boundary value problem . . . 5

2 siMesh object **8**

3 Python objects **10**
3.1 Loperator object . . . 10
3.1.1 Constructor . . . 11
3.2 Hoperator object . . . 12
3.2.1 Constructor . . . 12
3.2.2 Methods . . . 13
3.3 PDE object . . . 13
3.4 BVP object . . . 14
3.4.1 Constructor . . . 15
3.4.2 Main methods . . . 15
3.5 Finite element functions with Loperator object . . . 17
3.5.1 Notations on Ω_h . . . 17
3.5.2 Notations on Γ_h . . . 17
3.5.3 AssemblyP1 function . . . 18
3.5.4 **apply** function . . . 19

4 Scalar boundary value problems **20**
4.1 Poisson BVP's . . . 20
4.1.1 2D Poisson BVP with Dirichlet boundary conditions on the unit square . . . 20
4.1.2 2D Poisson BVP with mixed boundary conditions . . . 22
4.1.3 3D Poisson BVP with mixed boundary conditions . . . 23
4.1.4 1D BVP : just for fun . . . 25
4.2 Stationary convection-diffusion problem . . . 26
4.2.1 Stationary convection-diffusion problem in 2D . . . 26
4.2.2 Stationary convection-diffusion problem in 3D . . . 28
4.3 2D electrostatic BVPs . . . 30

5 Vector boundary value problems 33
5.1 Elasticity problem 33
5.1.1 General case ($d = 2, 3$) 33
5.1.2 2D example 34
5.1.3 3D example 36
5.2 Stationary heat with potential flow in 2D 37
5.2.1 Method 1 : split in three parts 39
5.2.2 Method 2 : have fun with $\mathcal{H}$-operators 41
5.3 Stationary heat with potential flow in 3D 43
5.3.1 Method 1 : split in three parts 46
5.3.2 Method 2 : have fun with $\mathcal{H}$-operators 47

Chapter 1

Generic Boundary Value Problems

The notations of [?] are employed in this section and extended to the vector case.

1.1 Scalar boundary value problem

Let Ω be a bounded open subset of $\mathbb{R}^d$, $d \geqslant 1$. The boundary of Ω is denoted by Γ.

We denote by $\mathcal{L}_{\mathbb{A},\boldsymbol{b},\boldsymbol{c},a_0} = \mathcal{L} : \mathrm{H}^2(\Omega) \longrightarrow L^2(\Omega)$ the second order linear differential operator acting on *scalar fields* defined, $\forall u \in \mathrm{H}^2(\Omega)$, by

$$\mathcal{L}_{\mathbb{A},\boldsymbol{b},\boldsymbol{c},a_0}(u) \stackrel{\text{def}}{=} -\operatorname{div}(\mathbb{A}\boldsymbol{\nabla} u) + \operatorname{div}(\boldsymbol{b}u) + \langle \boldsymbol{\nabla} u, \boldsymbol{c}\rangle + a_0 u \tag{1.1}$$

where $\mathbb{A} \in (L^\infty(\Omega))^{d\times d}$, $\boldsymbol{b} \in (L^\infty(\Omega))^d$, $\boldsymbol{c} \in (L^\infty(\Omega))^d$ and $a_0 \in L^\infty(\Omega)$ are given functions and $\langle\cdot,\cdot\rangle$ is the usual scalar product in $\mathbb{R}^d$. We use the same notations as in the chapter 6 of [?] and we note that we can omit either $\operatorname{div}(\boldsymbol{b}u)$ or $\langle \boldsymbol{\nabla} u, \boldsymbol{c}\rangle$ if $\boldsymbol{b}$ and $\boldsymbol{c}$ are sufficiently regular functions. We keep both terms with $\boldsymbol{b}$ and $\boldsymbol{c}$ to deal with more boundary conditions. It should be also noted that it is important to preserve the two terms $\boldsymbol{b}$ and $\boldsymbol{c}$ in the generic formulation to enable a greater flexibility in the choice of the boundary conditions.

Let Γ^D, Γ^R be open subsets of Γ, possibly empty and $f \in L^2(\Omega)$, $g^D \in \mathrm{H}^{1/2}(\Gamma^D)$, $g^R \in L^2(\Gamma^R)$, $a^R \in L^\infty(\Gamma^R)$ be given data.

A *scalar* boundary value problem is given by

Scalar BVP 1 : generic problem

Find $u \in \mathrm{H}^2(\Omega)$ such that

$$\mathcal{L}(u) = f \quad \text{in } \Omega, \tag{1.2}$$

$$u = g^D \quad \text{on } \Gamma^D, \tag{1.3}$$

$$\frac{\partial u}{\partial n_{\mathcal{L}}} + a^R u = g^R \quad \text{on } \Gamma^R. \tag{1.4}$$

The **conormal derivative** of u is defined by

$$\frac{\partial u}{\partial n_{\mathcal{L}}} \stackrel{\text{def}}{=} \langle \mathbb{A}\boldsymbol{\nabla} u, \boldsymbol{n}\rangle - \langle \boldsymbol{b}u, \boldsymbol{n}\rangle \tag{1.5}$$

The boundary conditions (1.3) and (1.4) are respectively **Dirichlet** and **Robin** boundary conditions. **Neumann** boundary conditions are particular Robin boundary conditions with $a^R \equiv 0$.

To have an outline of the FC-VFEM$\mathbb{P}_1$ package, a first and simple problem is quickly present. Explanations will be given in next sections.

The problem to solve is the Laplace problem for a condenser.

Usual BVP 1 : 2D condenser problem

Find $u \in \mathrm{H}^2(\Omega)$ such that

$$\begin{aligned} -\Delta u &= 0 \ \text{ in } \Omega \subset \mathbb{R}^2, & (1.6)\\ u &= 0 \ \text{ on } \Gamma_1, & (1.7)\\ u &= -12 \ \text{ on } \Gamma_{98}, & (1.8)\\ u &= 12 \ \text{ on } \Gamma_{99}, & (1.9) \end{aligned}$$

where Ω and its boundaries are given in Figure 1.1.

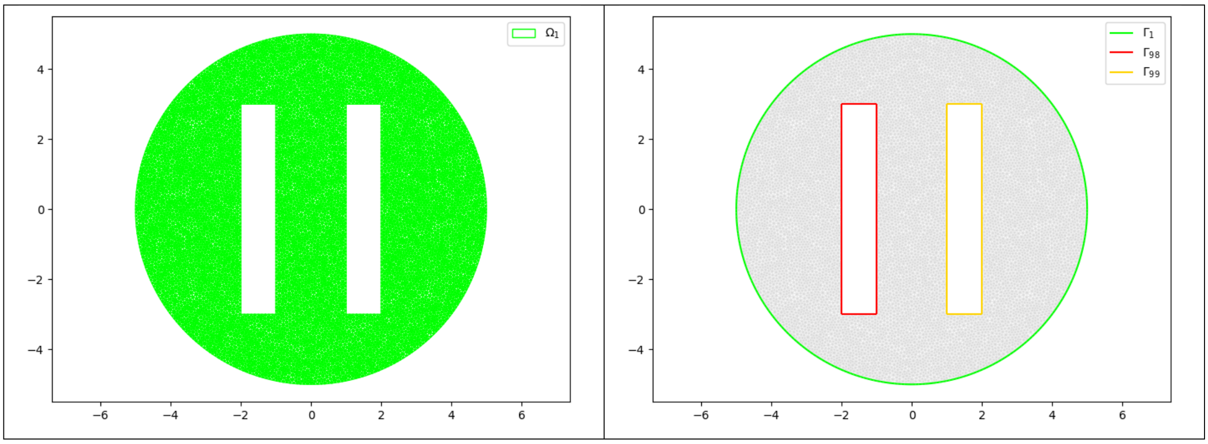


Figure 1.1: 2D condenser mesh and boundaries (left) and numerical solution (right)

The problem (1.6)-(1.9) can be equivalently expressed as the scalar BVP (1.2)-(1.4) :

Scalar BVP 2 : 2D condenser problem

Find $u \in \mathrm{H}^2(\Omega)$ such that

$$\begin{aligned} \mathcal{L}(u) &= f && \text{in } \Omega,\\ u &= g^D && \text{on } \Gamma^D = \Gamma_1 \cup \Gamma_{98} \cup \Gamma_{99}. \end{aligned}$$

where $\mathcal{L} := \mathcal{L}_{\mathbb{I},\mathbf{0},\mathbf{0},0}$, $f \equiv 0$, and

$$g^D := 0 \text{ on } \Gamma_1,\ g^D := -12 \text{ on } \Gamma_{98},\ g^D := +12 \text{ on } \Gamma_{99}$$

In Listing 27 a complete code is given to solve this problem.

```
meshfile=gmsh.buildmesh2d('condenser',10) # generate mesh
Th=siMesh(meshfile) # read mesh
Lop=Loperator(dim=2,d=2,A=[[1,0],[0,1]])
pde=PDE(Op=Lop)
bvp=BVP(Th,pde=pde)
bvp.setDirichlet( 1, 0.)
bvp.setDirichlet( 98, -12.)
bvp.setDirichlet( 99, +12.)
u=bvp.solve();
# Graphic parts
plt.figure(1)
siplt.plotmesh(Th,legend=True)
set_axes_equal()
plt.figure(2)
siplt.plotmesh(Th,color='LightGray',alpha=0.3)
siplt.plotmesh(Th,d=1,legend=True)
```

```
set_axes_equal()
plt.figure(3)
siplt.plot(Th,u)
plt.colorbar(label='u')
set_axes_equal()
plt.figure(4)
siplt.plotiso(Th,u,contours=15)
plt.colorbar(label='u')
siplt.plotmesh(Th,color='LightGray',alpha=0.3)
plt.axis('off');set_axes_equal()
```

Listing 1.1: Complete Python code to solve the 2D condenser problem with graphical representations

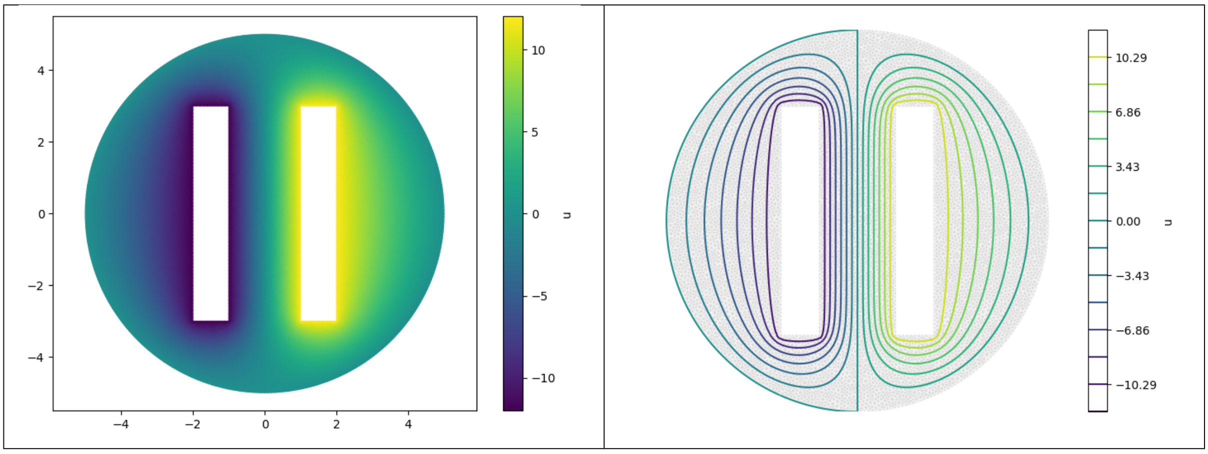


Figure 1.2: 2D condenser numerical solution

1.2 Vector boundary value problem

Let $m \geqslant 1$ and $\mathcal{H}$ be the m-by-m matrix of second order linear differential operators defined by

$$\left\{\begin{array}{llll}\mathcal{H} & : & \left(\mathrm{H}^2(\Omega)\right)^m & \longrightarrow \left(L^2(\Omega)\right)^m \\ & & \boldsymbol{u}=(\boldsymbol{u}_1,\ldots,\boldsymbol{u}_m) & \longmapsto \boldsymbol{f}=(\boldsymbol{f}_1,\ldots,\boldsymbol{f}_m) \stackrel{\mathsf{def}}{=} \mathcal{H}(\boldsymbol{u})\end{array}\right. \tag{1.10}$$

where

$$\boldsymbol{f}_\alpha = \sum_{\beta=1}^{m} \mathcal{H}_{\alpha,\beta}(\boldsymbol{u}_\beta), \quad \forall \alpha \in [\![1,m]\!], \tag{1.11}$$

with, for all $(\alpha,\beta) \in [\![1,m]\!]^2$,

$$\mathcal{H}_{\alpha,\beta} \stackrel{\mathsf{def}}{=} \mathcal{L}_{\mathbb{A}^{\alpha,\beta},\boldsymbol{b}^{\alpha,\beta},\boldsymbol{c}^{\alpha,\beta},a_0^{\alpha,\beta}} \tag{1.12}$$

and $\mathbb{A}^{\alpha,\beta} \in (L^\infty(\Omega))^{d\times d}$, $\boldsymbol{b}^{\alpha,\beta} \in (L^\infty(\Omega))^d$, $\boldsymbol{c}^{\alpha,\beta} \in (L^\infty(\Omega))^d$ and $a_0^{\alpha,\beta} \in L^\infty(\Omega)$ are given functions. We can also write in matrix form

$$\mathcal{H}(\boldsymbol{u}) = \begin{pmatrix} \mathcal{L}_{\mathbb{A}^{1,1},\boldsymbol{b}^{1,1},\boldsymbol{c}^{1,1},a_0^{1,1}} & \cdots & \mathcal{L}_{\mathbb{A}^{1,m},\boldsymbol{b}^{1,m},\boldsymbol{c}^{1,m},a_0^{1,m}} \\ \vdots & \ddots & \vdots \\ \mathcal{L}_{\mathbb{A}^{m,1},\boldsymbol{b}^{m,1},\boldsymbol{c}^{m,1},a_0^{m,1}} & \cdots & \mathcal{L}_{\mathbb{A}^{m,m},\boldsymbol{b}^{m,m},\boldsymbol{c}^{m,m},a_0^{m,m}} \end{pmatrix} \begin{pmatrix} \boldsymbol{u}_1 \\ \vdots \\ \boldsymbol{u}_m \end{pmatrix}. \tag{1.13}$$

We remark that the $\mathcal{H}$ operator for $m=1$ is equivalent to the $\mathcal{L}$ operator.

For $\alpha \in [\![1,m]\!]$, we define Γ_α^D and Γ_α^R as open subsets of Γ, possibly empty, such that $\Gamma_\alpha^D \cap \Gamma_\alpha^R = \emptyset$. Let $\boldsymbol{f} \in (L^2(\Omega))^m$, $g_\alpha^D \in \mathrm{H}^{1/2}(\Gamma_\alpha^D)$, $g_\alpha^R \in L^2(\Gamma_\alpha^R)$, $a_\alpha^R \in L^\infty(\Gamma_\alpha^R)$ be given data.

A *vector* boundary value problem is given by

Vector BVP 1 : generic problem

Find $\boldsymbol{u} = (\boldsymbol{u}_1, \dots, \boldsymbol{u}_m) \in (\mathrm{H}^2(\Omega))^m$ such that

$$\mathcal{H}(\boldsymbol{u}) = \boldsymbol{f} \quad \text{in } \Omega, \tag{1.14}$$
$$\boldsymbol{u}_\alpha = g_\alpha^D \quad \text{on } \Gamma_\alpha^D,\ \forall \alpha \in [\![1, m]\!], \tag{1.15}$$
$$\frac{\partial \boldsymbol{u}}{\partial n_{\mathcal{H}_\alpha}} + a_\alpha^R \boldsymbol{u}_\alpha = g_\alpha^R \quad \text{on } \Gamma_\alpha^R,\ \forall \alpha \in [\![1, m]\!], \tag{1.16}$$

where the α-th component of the **conormal derivative** of $\boldsymbol{u}$ is defined by

$$\frac{\partial \boldsymbol{u}}{\partial n_{\mathcal{H}_\alpha}} \stackrel{\text{def}}{=} \sum_{\beta=1}^{m} \frac{\partial \boldsymbol{u}_\beta}{\partial n_{\mathcal{H}_{\alpha,\beta}}} = \sum_{\beta=1}^{m} \left(\left\langle \mathbb{A}^{\alpha,\beta} \boldsymbol{\nabla} \boldsymbol{u}_\beta, \boldsymbol{n} \right\rangle - \left\langle \boldsymbol{b}^{\alpha,\beta} \boldsymbol{u}_\beta, \boldsymbol{n} \right\rangle \right). \tag{1.17}$$

The boundary conditions (1.16) are the **Robin** boundary conditions and (1.15) is the **Dirichlet** boundary condition. The **Neumann** boundary conditions are particular Robin boundary conditions with $a_\alpha^R \equiv 0$.

In this problem, we may consider on a given boundary some conditions which can vary depending on the component. For example we may have a Robin boundary condition satisfying $\frac{\partial \boldsymbol{u}}{\partial n_{\mathcal{H}_1}} + a_1^R \boldsymbol{u}_1 = g_1^R$ and a Dirichlet one with $\boldsymbol{u}_2 = g_2^D$.

To have an outline of the FC-VFEM$\mathbb{P}_1$ package, a second and simple problem is quickly present.

Usual vector BVP 1 : 2D simple vector problem

Find $\boldsymbol{u} = (u_1, u_2) \in (\mathrm{H}^2(\Omega))^2$ such that

$$-\Delta u_1 + u_2 = 0 \quad \text{in } \Omega \subset \mathbb{R}^2, \tag{1.18}$$
$$-\Delta u_2 + u_1 = 0 \quad \text{in } \Omega \subset \mathbb{R}^2, \tag{1.19}$$
$$(u_1, u_2) = (0, 0) \quad \text{on } \Gamma_1, \tag{1.20}$$
$$(u_1, u_2) = (-12., +12.) \quad \text{on } \Gamma_{98}, \tag{1.21}$$
$$(u_1, u_2) = (+12., -12.) \quad \text{on } \Gamma_{99}, \tag{1.22}$$

where Ω and its boundaries are given in Figure 1.1.

The problem (1.18)-(1.22) can be equivalently expressed as the vector BVP (1.2)-(1.4) :

Vector BVP 2 : 2D simple vector problem

Find $\boldsymbol{u} = (u_1, u_2) \in (\mathrm{H}^2(\Omega))^2$ such that

$$\mathcal{H}(\boldsymbol{u}) = \boldsymbol{f} \quad \text{in } \Omega,$$
$$u_1 = g_1^D \quad \text{on } \Gamma^D = \Gamma_1 \cup \Gamma_{98} \cup \Gamma_{99},$$
$$u_2 = g_2^D \quad \text{on } \Gamma^D = \Gamma_1 \cup \Gamma_{98} \cup \Gamma_{99},$$

where

$$\mathcal{H} := \begin{pmatrix} \mathcal{L}_{\mathbb{I},\boldsymbol{O},\boldsymbol{O},0} & \mathcal{L}_{\mathbb{O},\boldsymbol{O},\boldsymbol{O},1} \\ \mathcal{L}_{\mathbb{O},\boldsymbol{O},\boldsymbol{O},1} & \mathcal{L}_{\mathbb{I},\boldsymbol{O},\boldsymbol{O},0} \end{pmatrix}, \text{ as } \mathcal{H}\begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \begin{pmatrix} -\Delta & 1 \\ 1 & -\Delta \end{pmatrix}\begin{pmatrix} u_1 \\ u_2 \end{pmatrix}$$
$$f \equiv 0,$$

and

$$g_1^D = g_2^D := 0 \text{ on } \Gamma_1,\ g_1^D := -12,\ g_2^D := +12 \text{ on } \Gamma_{98},\ g_1^D := +12,\ g_2^D := -12 \text{ on } \Gamma_{99}$$

In Listing 21 a complete code is given to solve this problem. Numerical solutions are given in Figure 1.3.

```
meshfile=gmsh.buildmesh2d('condenser',10); # generate mesh
Th=siMesh(meshfile) # read mesh
Hop1=Loperator(dim=2,A=[[1,None],[None,1]])
```

```
Hop2=Loperator(dim=2,a0=1)
Hop=Hoperator(dim=2,m=2,H=[[Hop1,Hop2],[Hop2,Hop1]])
pde=PDE(Op=Hop)
bvp=BVP(Th,pde=pde)
bvp.setDirichlet( 1, 0,comps=[0,1])
bvp.setDirichlet( 98, [-12,+12],comps=[0,1]);
bvp.setDirichlet( 99, [+12,-12],comps=[0,1]);
U=bvp.solve(split=True)
# Graphic parts
plt.figure(1)
siplt.plot(Th,U[0])
plt.axis('off');set_axes_equal()
plt.colorbar(label='$u_1$',orientation='horizontal')
plt.figure(2)
siplt.plot(Th,U[1])
plt.axis('off');set_axes_equal()
plt.colorbar(label='$u_2$',orientation='horizontal')
```

Listing 1.2: Complete Python code to solve the funny 2D vector problem with graphical representations

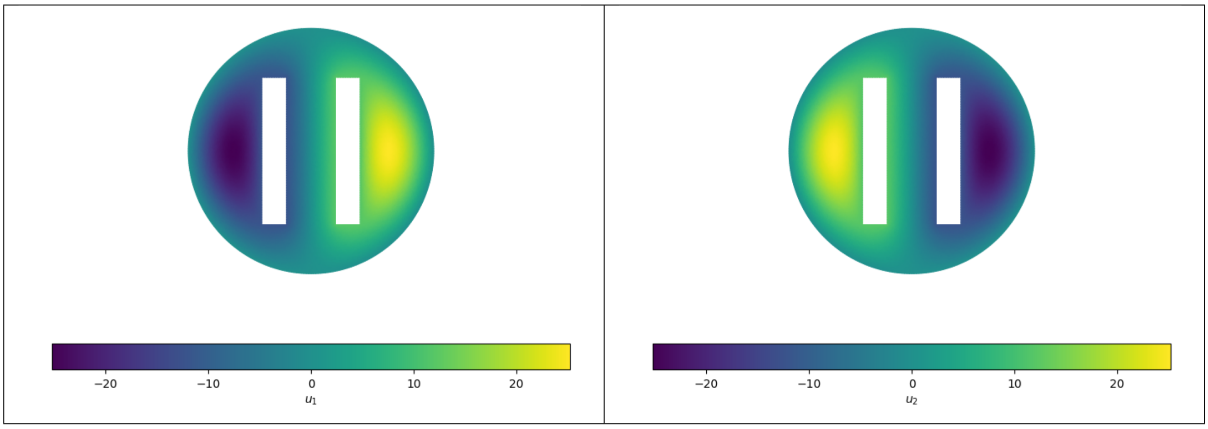


Figure 1.3: Funny vector BVP, u_1 numerical solution (left) and u_2 numerical solution (right)

Obviously, more complex problems will be studied in section **??** and complete explanations on the code will be given in next sections.

In the following of the report we will solve by a $\mathbb{P}^1$-Lagrange finite element method *scalar* B.V.P. (1.2) to (1.4) and *vector* B.V.P. (1.14) to (1.16) without additional restrictive assumption.

Chapter 2

siMesh object

The siMesh object is defined in the FC-SIMESH Python package

`http://www.math.univ-paris13.fr/~cuvelier/software/fc-simesh-Python.html`

An user's guide is also provided on this web page.

As 2d example, we use in this report the mesh obtained from `square4holes6dom.geo` by using the code given in Listing 2.1 and represented in Figure 2.1.

Listing 2.1: Python code to get a 2D mesh

```
from fc_simesh.siMesh import siMesh
from fc_oogmsh import gmsh
from fc_vfemp1.sys import get_geo
(geodir, geofile)=get_geo(2,2,'square4holes6dom')
meshfile=gmsh.buildmesh(2,geodir+'/'+geofile,50,verbose=0)
Th=siMesh(meshfile)
print('Th -> '+str(Th));
```

Output

```
Th -> siMesh object
     d : 2
   dim : 2
   nq  : 13258
   nme : 25994
   sTh : list of 16 siMeshElt
  nsTh : 16
sThsimp : (16,) ndarray
   [1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2]
 sThlab : (16,) ndarray
   [ 1  2  3  4  5  6  7  8 10 20 2  4  6  8 10 20]
```

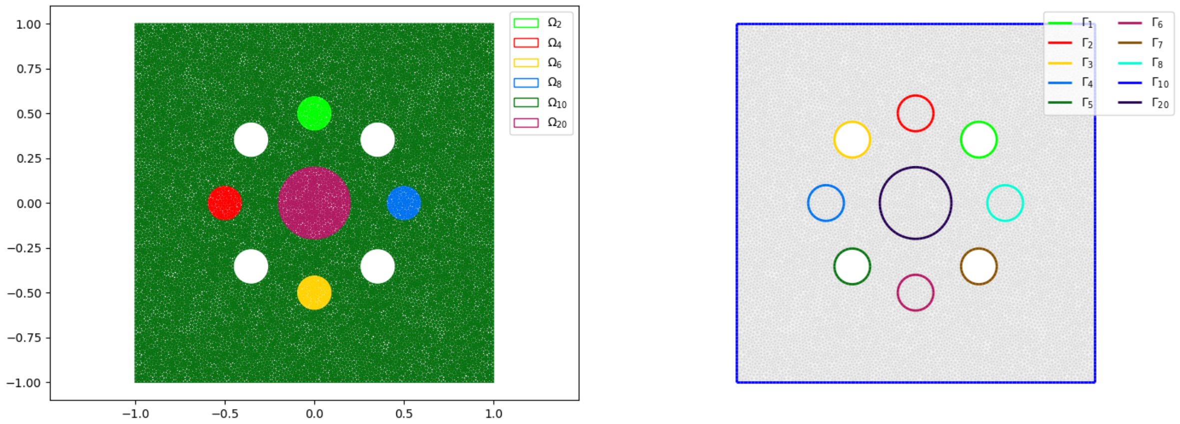


Figure 2.1: Mesh from `square4holes6dom.geo`, domains representation (left) and boundaries (right)

From the 2d mesh example given in Listing 2.1 and Figure 2.1 we have

$$\Omega_h = \bigcup_{l \in \text{labs2}} \Omega_h^l \qquad \text{with} \quad \text{labs2} = \{2, 4, 6, 8, 10, 20\} \tag{2.1}$$

$$\Gamma_h \overset{\text{def}}{=} \partial\Omega_h = \bigcup_{l \in \text{labs1}} \Gamma_h^l \qquad \text{with} \quad \text{labs1} = \{1, 3, 5, 7, 10, 20\} \tag{2.2}$$

where each one of the Ω_h^l is a 2-simplicial elementary mesh and each one of the Γ_h^l is a 1-simplicial elementary mesh. With the siMesh object `Th` , we easily can obtain each one of these elementary meshes. For example, we have

$$\Omega_h^8 \leftarrow \text{Th.sTh[Th.find(2,8)]} \qquad\qquad \Gamma_h^{10} \leftarrow \text{Th.sTh[Th.find(1,10)]}$$

Chapter 3

Python objects

3.1 Loperator object

The object `Loperator` is used to create the operator $\mathcal{L}_{\mathbb{A},\boldsymbol{b},\boldsymbol{c},a_0}$ defined in (1.1).

The `Loperator` main properties are

Properties of `Loperator` object

`dim`	:	integer, space dimension.
`A`	:	a `dim` -by- `dim` list. Used to store the $\mathbb{A}$ functions such that A[i−1][j−1] $\leftarrow \mathbb{A}_{i,j}$.
`b`	:	list with `dim` elements. Used to store the $\boldsymbol{b}$ functions such that b[i−1] $\leftarrow \boldsymbol{b}_i$.
`c`	:	list with `dim` elements. Used to store the $\boldsymbol{c}$ functions such that c[i−1] $\leftarrow \boldsymbol{c}_i$.
`a0`	:	 Used to store the a_0 function such that a0 $\leftarrow a_0$.

To set all the functions contain in this operator (i.e. function from $\mathbb{R}^{\text{dim}}$ to $\mathbb{R}$) we accept in Python various type of datas collectively known as *operator data function* and given by

Definition 3.1: ***operator data function***

In Python, we say that a data is an *operator data function* if its

- a scalar (for constant function),
- a **`def`** function,
- a **`lambda`** function,

- a numpy array (when used with a mesh),
- a `None` value for the 0 function.

3.1.1 Constructor

Its contructor is

```
obj=Loperator(**kwargs)
```

Description

`obj=Loperator(key=value,...)`. The key could be

- `dim` : to set the space dimension (default 2),
- `A` : to set the matrix-valued function $\mathbb{A}$ (default is a list of `dim` lists of `dim` `None` if `fill` is `True` otherwise default is `None`),
- `b` : to set the vector-valued function $\boldsymbol{b}$ (default is a list of `dim` `None` otherwise default is `None`),
- `c` : to set the vector-valued function $\boldsymbol{c}$ (default is a list of `dim` `None` otherwise default is `None`),
- `a0` : to set the function a_0 (default `None`),
- `fill` : if `True` fill the default data with a `dim` -by- `dim` *matrix* of `None` for field `A` and with a *vector* of `None` for fields `b` and `c` (default `False`).

Samples

$-\Delta u := \mathcal{L}_{\mathbb{I},\boldsymbol{O},\boldsymbol{O},0}(u)$

in $\mathbb{R}$ `Lop=Loperator(dim=1,A=[[1]])`
in $\mathbb{R}^2$ `Lop=Loperator(dim=2,A=[[1,None],[None,1]])`
in $\mathbb{R}^3$ `Lop=Loperator(dim=3,A=[[1,None,None],[None,1,None],[None,None,1]])`
$\vdots$

$-\Delta u + u := \mathcal{L}_{\mathbb{I},\boldsymbol{O},\boldsymbol{O},1}(u)$

in $\mathbb{R}$ `Lop=Loperator(dim=1,A=[[1]],a0=1)`
in $\mathbb{R}^2$ `Lop=Loperator(dim=2,A=[[1,None],[None,1]],a0=1)`
in $\mathbb{R}^3$ `Lop=Loperator(dim=3,A=[[1,None,None],[None,1,None],[None,None,1]],a0=1)`
$\vdots$

In R^2, $-\Delta u + (1 + cos(x + y))u := \mathcal{L}_{\mathbb{I},\boldsymbol{O},\boldsymbol{O},(x,y)\mapsto(1+cos(x+y))}(u)$

```
Lop=Loperator(dim=2,A=[[1,None],[None,1]],a0=lambda x,y:1+np.cos(x+y))
```
or
```
Lop=Loperator(dim=2,A=[[1,None],[None,1]],a0=lambda X:1+np.cos(X[0]+X[1]))
```

In R^3, let $\alpha : (x, y, z) \mapsto 1 + x^2 + y^2 + z^2$, $\beta : (x, y, z) \mapsto 1 + x^2$, $\boldsymbol{V} = \begin{pmatrix} (x, y, z) \mapsto x + y + z \\ 1 \\ (x, y, z) \mapsto x * y * z \end{pmatrix}$ and

$\mathbb{A} = \begin{pmatrix} \alpha & 0 & 0 \\ 0 & \alpha & 0 \\ 0 & 0 & \alpha \end{pmatrix}$. Then we have

$$-\operatorname{div}(\alpha \boldsymbol{\nabla} u) + \langle \boldsymbol{V}, \boldsymbol{\nabla} u\rangle + \beta u := \mathcal{L}_{\mathbb{A},\boldsymbol{0},\boldsymbol{V},\beta}(u).$$

The associated python code is given in Listing 3.1

Listing 3.1: sample

```
from fc_vfempl.operators import Loperator
af=lambda x,y,z: 1+x**2+y**2+z**2
Vx=lambda x,y,z: x+y+z; Vy=1; Vz=lambda x,y,z: x*y*z
beta=lambda x,y,z: 1+x**2
Lop=Loperator(dim=3,A=[[af,None,None],[None,af,None],[None,None,af]], c=[Vx,Vy,Vz], a0=beta)
print(Lop)
```

Output

```
Loperator : (dim,d,order) = (3,3,2)
      A : [[Function, None, None], [None, Function, None], [None, None, Function]]
      b : None
      c : [Function, Scalar, Function]
     a0 : Function
```

3.2 Hoperator object

The object `Hoperator` is used to create the operator $\mathcal{H}$ defined in (1.10). Its main properties are

Properties of Hoperator object

dim	:	integer, space dimension.
m	:	integer
H	:	List of m-by-m elements. Used to store the $\mathcal{H}$ operators such that H[i−1][j−1] $\leftarrow \mathcal{H}_{i,j}$, $\forall i,j \in [\![1,m]\!]$. Each element contains a Loperator object or a None value.

3.2.1 Constructor

Its contructor are

```
obj=Hoperator()
obj=Hoperator(dim=..., m=...)
obj=Hoperator(dim=..., m=..., H=...)
```

Description

`obj=Hoperator()` create an empty/null operator with `dim=2` and `m=2` .

`obj=Hopertor(dim=..., m=...)` create an empty/null operator with the given dimensions `dim` and `m` .

`obj=Hopertor(dim=...,m=..., H=...)` create an operator with a given `H` .

Samples

In $\mathbb{R}^2$, with $\boldsymbol{u} = (u_1, u_2)$ the operator $\mathcal{H}$ defined by

$$\mathcal{H}(\boldsymbol{u}) \stackrel{\text{def}}{=} \begin{pmatrix} -\Delta u_1 + u_2 \\ u_1 - \Delta u_2 \end{pmatrix}$$

could be written as

$$\mathcal{H}\begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \begin{pmatrix} -\Delta & 1 \\ 1 & -\Delta \end{pmatrix}\begin{pmatrix} u_1 \\ u_2 \end{pmatrix}$$

and then

$$\mathcal{H} = \begin{pmatrix} \mathcal{L}_{\mathbb{I},\boldsymbol{O},\boldsymbol{O},0} & \mathcal{L}_{\mathbb{O},\boldsymbol{O},\boldsymbol{O},1} \\ \mathcal{L}_{\mathbb{O},\boldsymbol{O},\boldsymbol{O},1} & \mathcal{L}_{\mathbb{I},\boldsymbol{O},\boldsymbol{O},0} \end{pmatrix}$$

Listing 3.2: sample

```
from fc_vfemp1.operators import Loperator,Hoperator
Lop1=Loperator(dim=2,A=[[1,None],[None,1]])
Lop2=Loperator(dim=2,a0=1)
Hop1=Hoperator(dim=2,m=2,H=[[Lop1,Lop2],[Lop2,Lop1]])
print('*** Hop1 -> '+str(Hop1))
Hop2=Hoperator(dim=2,m=2)
Hop2.set(0,0,Lop1);Hop2.set(0,1,Lop2)
Hop2.set(1,0,Lop2);Hop2.set(1,1,Lop1)
print('*** Hop2 -> '+str(Hop2))
```

Output

```
*** Hop1 -> Hoperator :
  (dim,d,m) : (2,2,2)
      order : 2
    H[0][0] : Loperator : (dim,d,order) = (2,2,2)
           A : [[Scalar, None], [None, Scalar]]
    H[0][1] : Loperator : (dim,d,order) = (2,2,0)
          a0 : Scalar
    H[1][0] : Loperator : (dim,d,order) = (2,2,0)
          a0 : Scalar
    H[1][1] : Loperator : (dim,d,order) = (2,2,2)
           A : [[Scalar, None], [None, Scalar]]
*** Hop2 -> Hoperator :
  (dim,d,m) : (2,2,2)
      order : 2
    H[0][0] : Loperator : (dim,d,order) = (2,2,2)
           A : [[Scalar, None], [None, Scalar]]
    H[0][1] : Loperator : (dim,d,order) = (2,2,0)
          a0 : Scalar
    H[1][0] : Loperator : (dim,d,order) = (2,2,0)
          a0 : Scalar
    H[1][1] : Loperator : (dim,d,order) = (2,2,2)
           A : [[Scalar, None], [None, Scalar]]
```

3.2.2 Methods

set function

zeros function

opStiffElas function

3.3 PDE object

This object is used to store the scalar PDE (1.2) or the vector PDE (1.14):

$$\mathcal{L}(u) = f \quad \text{or} \quad \mathcal{H}(\boldsymbol{u}) = \boldsymbol{f}$$

acting on d-dimensional submanifold of $\mathbb{R}^{\text{dim}}$. For example with dim $= 3$, a 2-dimensional submanifold is a surface, a 1-dimensional submanifold is a curve and a 0-dimensional submanifold is a point. Its main properties are

Properties of PDE object

dim	:	integer, space dimension.
d	:	integer, submanifold dimension.
m	:	integer
Op	:	Loperator or Hoperator object.
f	:	(list of) *operator data function* or None. Used to store the right-hand side of the PDE. If Op is an Loperator object then f is an *operator data function*. If Op is an Hoperator object then f is a list of Op.m *operator data function* or None value.

Its contructor are

```
obj=PDE()
obj=PDE(dim=..., m=..., Op=..., f=...)
```

Description

`obj=PDE()` create an *empty* PDE object with `dim=2` and `m=1`

`obj=PDE(dim=dval)` create an *empty* PDE object with `dim=dval` and `m=1`

`obj=PDE(m=mval)` create an *empty* PDE object with `dim=2` and `m=mval`

`obj=PDE(Op=op)` create the PDE object with $f \equiv 0,$: i.e. `Op(u)=0` with `dim=Op.dim` and `m=1` if fclstop is a `Loperator` object and `m=op.m` if `op` is an `Hoperator` object.

`obj=PDE(Op=op,f=fun)` create the PDE `Op(u)=f`. If `Op` is an `Hoperator` object then `f` must be a cell array of length `Hoperator.m` .

Samples

In $\mathbb{R}^2$, $-\Delta u + u = f$, with $f(x, y) = xy^2$

Listing 3.3: Test

```python
from fc_vfemp1.operators import Loperator
from fc_vfemp1.BVP import PDE
Lop=Loperator(dim=2,A=[[1,None],[None,1]],a0=1)
g=lambda x,y: x*y**2
pde=PDE(Op=Lop,f=g)
print(pde)
```

Output

```
PDE object : (dim,m) = (2,1)
    Op : Loperator : (dim,d,order) = (2,2,2)
         A : [[Scalar, None], [None, Scalar]]
         b : None
         c : None
        a0 : Scalar
     f : <function <lambda> at 0x2ad86db52e18>
 delta : [ 0.]
```

3.4 BVP object

The object `BVP` is used to create a scalar boundary value problem (1.2)-(1.4) or a vector boundary value problem (1.14)-(1.16). The usage of this object is strongly correlated with good comprehension of the FC-SIMESH package and more particularly with the `siMesh` object.

The properties of the object `BVP` are

Properties of `BVP` object

Th	:	a `siMesh` object
dim	:	integer, space dimension (equal to `Th.dim`).
d	:	integer, (equal to `Th.d`).
m	:	integer, system of `m` PDE's.
pdes	:	list of `Th.nsTh` `PDE` objects. Used to store the PDE associated with each submesh Th.sTh[i]. If pdes[i] is `None` then there is no PDE defined on Th.sTh[i].

3.4.1 Constructor

Its contructor are

```
obj=BVP(Th)
obj=BVP(Th,pde=...,labels=...)
```

Description

`obj=BVP(Th)` create a `BVP` object with no PDE's defined,

`obj=BVP(Th,pde=pde)` create a `BVP` object with PDE's defined by `pde` object on all submeshes of index Th.find(pde.d) i.e. on all submeshes such that Th.sTh[i].d==pde.d. By default, homogeneous Neumann boundary conditions are set on all *boundaries*.

`obj=BVP(Th,pde=pde,labels=labs)` similar to previous one except among the selected objects are choosen those with label (Th.sTh[i]. label) in `labs` array/list. By default, homogeneous Neumann boundary conditions are set on all *boundaries*.

3.4.2 Main methods

Let `bvp` be a `BVP` object.

setPDE function

```
bvp.setPDE(pde)
bvp.setPDE(pde,labels=..., d=...)
```

Description

`bvp.setPDE(pde)` associated the `pde` object with the all the d-dimensional elementary meshes where d is `bvp.Th.d` .

`bvp.setPDE(pde,labels=labs,d=dval)` associated the `pde` object with the `dval` -dimensional elementary meshes with labels in `labs` array/list.

setDirichlet function

```
bvp.setDirichlet(label,g)
bvp.setDirichlet(label,g,m=...)
```

Description

`bvp.setDirichlet(label,g)` for scalar B.V.P., sets Dirichlet boundary condition (1.3)

$$u = \texttt{g}, \quad \text{on } \Gamma_{\text{label}}$$

and for vector B.V.P., sets Dirichlet boundary condition (1.15)

$$u_\alpha = \texttt{g}[\alpha - 1], \forall \alpha \in [\![1, m[\![\text{ on } \Gamma_{\texttt{label}}.$$

`bvp.setDirichlet(label,g,comps=Lc)` for vector B.V.P., sets Dirichlet boundary condition : $\forall \alpha \in$ `Lc`, let i such that $\alpha =$ `Lc`$[i]$ then

$$u_\alpha = \texttt{g[i]}, \text{ on } \Gamma_{\texttt{label}}.$$

setRobin function

```
bvp.setRobin(label,gr)
bvp.setRobin(label,gr,ar=...,comps=...)
```

Description

`bvp.setRobin(label,gr)` for scalar B.V.P., sets Neumann boundary condition (i.e. Robin boundary condition (??) with $a^R = 0$)

$$\frac{\partial u}{\partial n_{\mathcal{L}}} = \text{gr}, \quad \text{on } \Gamma_{\text{label}}.$$

For vector B.V.P., sets Neumann boundary conditions (i.e. Robin boundary condition (??) with $a^R_\alpha = 0$)

$$\frac{\partial \boldsymbol{u}}{\partial n_{\mathcal{H}_\alpha}} = \text{gr}[\alpha - 1], \quad \forall \alpha \in [\![1, m]\!] \text{ on } \Gamma_{\text{label}}.$$

`bvp.setRobin(label,gr,ar=arfun)` for scalar B.V.P., sets Robin boundary condition (1.4)

$$\frac{\partial u}{\partial n_{\mathcal{L}}} + \text{arfun} \times u = \text{gr}, \quad \text{on } \Gamma_{\text{label}}.$$

For vector B.V.P., sets Robin boundary condition (1.16)

$$\frac{\partial \boldsymbol{u}}{\partial n_{\mathcal{H}_i}} + \text{arfun[i−1]}\boldsymbol{u}_i = \text{gr[i−1]}, \quad \forall i \in [\![1, m]\!] \text{ on } \Gamma_{\text{label}}.$$

`bvp.setRobin(label,gr,comps=Lc)` for vector B.V.P., sets Robin boundary condition (1.16) : $\forall \alpha \in$ `Lc`, let i such that $\alpha =$ `Lc` $[i]$ then the α-th components equation is

$$\frac{\partial \boldsymbol{u}}{\partial n_{\mathcal{H}_\alpha}} + \text{arfun[i]} \times \boldsymbol{u}_\alpha = \text{gr[i]}, \quad \text{on } \Gamma_{\text{label}}.$$

solve function

```
x=bvp.solve()
x=bvp.solve(key,value,...)
```

Description

`x=bvp.solve()` uses P_1-Lagrange finite elements method to solve the B.V.P. described by the `bvp` object.

`x=bvp.solve(key,value,...)`

- `'solver'` :
- `'split'` :
- `'local'` :
- `'perm'` :

3.5 Finite element functions with Loperator object

3.5.1 Notations on Ω_h

Let $\mathcal{D}_{\mathcal{L}} = \mathcal{D}_{\mathbb{A},\boldsymbol{b},\boldsymbol{c},a_0}$ be the first order bilinear differential operator acting on *scalar fields* associated to the $\mathcal{L}_{\mathbb{A},\boldsymbol{b},\boldsymbol{c},a_0}$ operator defined $\forall(u,v) \in (\mathrm{H}^1(\Omega))^2$ by

$$\mathcal{D}_{\mathcal{L}}(u,v) = \langle \mathbb{A}\,\boldsymbol{\nabla}\, u, \boldsymbol{\nabla}\, v\rangle - (u\,\langle \boldsymbol{b}, \boldsymbol{\nabla}\, v\rangle - v\,\langle \boldsymbol{\nabla}\, u, \boldsymbol{c}\rangle) + a_0 uv. \tag{3.1}$$

Let $\Omega_h = \bigcup_{l\in \text{ labs }} \Omega_h^l$ be a partition of Ω_h. We denote by

- $\widehat{\mathbb{D}}^{\mathcal{L}}(\Omega_h^l)$ the $\widehat{\mathrm{n}}_\mathrm{q}$-by-$\widehat{\mathrm{n}}_\mathrm{q}$ (local) matrix defined by

$$\widehat{\mathbb{D}}^{\mathcal{L}}_{i,j}(\Omega_h^l) = \int_{\Omega_h^l} \mathcal{D}_{\mathcal{L}}(\widehat{\varphi}_j, \widehat{\varphi}_i) d\mathrm{q} \tag{3.2}$$

  where $\widehat{\mathrm{n}}_\mathrm{q}$ is the number of vertices of Ω_h^l and $\{\widehat{\varphi}_i\}_{i\in[\![1,\widehat{\mathrm{n}}_\mathrm{q}]\!]}$ are the (local) $\mathbb{P}_1$-Lagrange basis functions on Ω_h^l.

- $\mathbb{D}^{\mathcal{L}}(\Omega_h^l)$ the n_q-by-n_q (global) matrix defined by

$$\mathbb{D}^{\mathcal{L}}_{i,j}(\Omega_h^l) = \int_{\Omega_h^l} \mathcal{D}_{\mathcal{L}}(\varphi_j, \varphi_i) d\mathrm{q} \tag{3.3}$$

  where n_q is the number of vertices of Ω_h and $\{\varphi_i\}_{i\in[\![1,\mathrm{n}_\mathrm{q}]\!]}$ are the (global) $\mathbb{P}_1$-Lagrange basis functions on Ω_h.

- $\mathbb{D}^{\mathcal{L}}(\Omega_h)$ the n_q-by-n_q matrix defined by

$$\mathbb{D}^{\mathcal{L}}_{i,j}(\Omega_h) = \int_{\Omega_h} \mathcal{D}_{\mathcal{L}}(\varphi_j, \varphi_i) d\mathrm{q} \tag{3.4}$$

  where n_q is the number of vertices of Ω_h and $\{\varphi_i\}_{i\in[\![1,\mathrm{n}_\mathrm{q}]\!]}$ are the (global) $\mathbb{P}_1$-Lagrange basis functions on Ω_h. We can remark that

$$\mathbb{D}^{\mathcal{L}}(\Omega_h) = \sum_{l\in \text{ labs }} \mathbb{D}^{\mathcal{L}}(\Omega_h^l)$$

3.5.2 Notations on Γ_h

Let $\mathcal{D}_{\mathcal{L}} = \mathcal{D}_{\mathbb{A},\boldsymbol{b},\boldsymbol{c},a_0}$ be the first order bilinear differential operator acting on *scalar fields* associated to the $\mathcal{L}_{\mathbb{A},\boldsymbol{b},\boldsymbol{c},a_0}$ operator defined $\forall(u,v) \in (\mathrm{H}^1(\Gamma))^2$ by

$$\mathcal{D}_{\mathcal{L}}(u,v) = \langle \mathbb{A}\,\boldsymbol{\nabla}_\Gamma\, u, \boldsymbol{\nabla}_\Gamma\, v\rangle - (u\,\langle \boldsymbol{b}, \boldsymbol{\nabla}_\Gamma\, v\rangle - v\,\langle \boldsymbol{\nabla}_\Gamma\, u, \boldsymbol{c}\rangle) + a_0 uv. \tag{3.5}$$

where $\mathbb{A}$....

Let $\Gamma_h = \bigcup_{l\in \text{ labs }} \Gamma_h^l$ be a partition of Γ_h. We denote by

- $\widehat{\mathbb{D}}^{\mathcal{L}}(\Gamma_h^l)$ the $\widehat{\mathrm{n}}_\mathrm{q}$-by-$\widehat{\mathrm{n}}_\mathrm{q}$ (local) matrix defined by

$$\widehat{\mathbb{D}}^{\mathcal{L}}_{i,j}(\Gamma_h^l) = \int_{\Gamma_h^l} \mathcal{D}_{\mathcal{L}}(\widehat{\varphi}_j, \widehat{\varphi}_i) d\mathrm{q} \tag{3.6}$$

  where $\widehat{\mathrm{n}}_\mathrm{q}$ is the number of vertices of Γ_h^l and $\{\widehat{\varphi}_i\}_{i\in[\![1,\widehat{\mathrm{n}}_\mathrm{q}]\!]}$ are the (local) $\mathbb{P}_1$-Lagrange basis functions on Γ_h^l.

- $\mathbb{D}^{\mathcal{L}}(\Gamma_h^l)$ the n_q-by-n_q (global) matrix defined by

$$\mathbb{D}^{\mathcal{L}}_{i,j}(\Gamma_h^l) = \int_{\Gamma_h^l} \mathcal{D}_{\mathcal{L}}(\varphi_j, \varphi_i) d\mathrm{q} \tag{3.7}$$

  where n_q is the number of vertices of Ω_h and $\{\varphi_i\}_{i\in[\![1,\mathrm{n}_\mathrm{q}]\!]}$ are the (global) $\mathbb{P}_1$-Lagrange basis functions on Ω_h.

- $\mathbb{D}^{\mathcal{L}}(\Gamma_h)$ the $\mathrm{n_q}$-by-$\mathrm{n_q}$ matrix defined by

$$\mathbb{D}^{\mathcal{L}}_{i,j}(\Gamma_h) = \int_{\Gamma_h} \mathcal{D}_{\mathcal{L}}(\varphi_j, \varphi_i) dq \tag{3.8}$$

where $\mathrm{n_q}$ is the number of vertices of Ω_h and $\{\varphi_i\}_{i\in[\![1,\mathrm{n_q}]\!]}$ are the (global) $\mathbb{P}_1$-Lagrange basis functions on Ω_h. We can remark that

$$\mathbb{D}^{\mathcal{L}}(\Gamma_h) = \sum_{l\in\ \mathrm{labs}} \mathbb{D}^{\mathcal{L}}(\Gamma_h^l)$$

3.5.3 AssemblyP1 function

Let `Th` be a siMesh object representing Ω_h and, at least, all its boundaries. Let `eTh` be a siMeshElt object obtained from the array `Th.sTh` (i.e. `eTh=Th.sTh[idx]`). Let `Lop` be the Loperator object representing $\mathcal{L}_{\mathbb{A},\boldsymbol{b},\boldsymbol{c},a_0}$.

`D=AssemblyP1(Th,Lop)` returns the `Th.nq` -by- `Th.nq` matrix $\mathbb{D}^{\mathcal{L}}(\Omega_h)$ defined in (3.8).

As example, we compute in Listing 3.4 the Mass matrix and the Stiffness matrix for the mesh obtain with Listing 2.1.

Listing 3.4: Examples of usage of the function `AssemblyP1`

```
from fc_vfemp1.operators import Loperator
from fc_vfemp1.FEM import AssemblyP1
LopMass=Loperator(dim=2,a0=1)
print('*** Compute mass matrix')
Mass=AssemblyP1(Th,LopMass)
print('Mass -> '+Mass.__repr__())
print('*** Compute stiffness matrix')
LopStiff=Loperator(dim=2,A=[[1,None],[None,1]])
Stiff=AssemblyP1(Th,LopStiff)
print('Stiff -> '+Stiff.__repr__())
```

Output

```
*** Compute mass matrix
Mass -> <13258x13258 sparse matrix of type '<class 'numpy.float64'>'
        with 91768 stored elements in Compressed Sparse Column format>
*** Compute stiffness matrix
Stiff -> <13258x13258 sparse matrix of type '<class 'numpy.float64'>'
        with 91768 stored elements in Compressed Sparse Column format>
```

`D=AssemblyP1(Th,Lop,labels=labs)` returns the `Th.nq` -by- `Th.nq` matrix $\mathbb{D}^{\mathcal{L}}(\Omega_h^{\mathrm{labs}})$ where

$$\Omega_h^{\mathrm{labs}} = \bigcup_{l\in\ \mathrm{labs}} \Omega_h^l \subset \Omega_h$$

and we also have

$$\mathbb{D}^{\mathcal{L}}(\Omega_h^{\mathrm{labs}}) = \sum_{l\in\ \mathrm{labs}} \mathbb{D}^{\mathcal{L}}(\Omega_h^l),\ \text{with}\ \mathbb{D}^{\mathcal{L}}_{i,j}(\Omega_h^l) = \int_{\Omega_h^l} \mathcal{D}_{\mathcal{L}}(\varphi_j, \varphi_i) dq \forall (i,j) \in [\![1,\mathrm{n_q}]\!]^2$$

Listing 3.5: function `AssemblyP1` with labels option

```
Th=siMesh(meshfile)
from fc_vfemp1.operators import Loperator
from fc_vfemp1.FEM import AssemblyP1
Lop=Loperator(dim=2,A=[[1,None],[None,1]],a0=lambda x,y:1+x**2)
M = AssemblyP1(Th,Lop)
Ma= AssemblyP1(Th,Lop,labels=[2,4,6,8])
Mb= AssemblyP1(Th,Lop,labels=[10,20])
E = M-(Ma+Mb)
print('E -> '+E.__repr__())
```

Output

```
E -> <13258x13258 sparse matrix of type '<class 'numpy.float64'>'
        with 0 stored elements in Compressed Sparse Column format>
```

LoM,LtoG=AssemblyP1(Th,Lop,local=True)

returns a list of sparse matrix LoM and a list of numpy array LtoG . With idx=Th.find(Th.d) , we have

LoM[i]=AssemblyP1(Th.sTh[idx[i]],Lop)

and

LtoG[i]=Th.sTh[idx[i]].toGlobal .

LoM,LtoG=AssemblyP1(Th,Lop,local=True,labels=labs)

D=AssemblyP1(Th,Lop,d=dvalue)

D=AssemblyP1(Th,Lop,d=dvalue,labels=labs)

LoM,LtoG=AssemblyP1(Th,Lop,d=dvalue,local=True)

LoM,LtoG=AssemblyP1(Th,Lop,d=dvalue,local=True,labels=labs)

3.5.4 apply function

This function can be used to numerically compute $\langle \boldsymbol{c}, \boldsymbol{\nabla} u\rangle + a_0 u := \mathcal{L}_{\mathbb{O},\boldsymbol{O},\boldsymbol{c},a0}(u)$ on a given mesh. This method only works if the A and b properties of the operator are None .

U=**apply**(Th,Lop,u) returns the numpy array with Th.nq ...

Chapter 4

Scalar boundary value problems

4.1 Poisson BVP's

The generic problem to solve is the following

Usual BVP 2 : Poisson problem

Find $u \in \mathrm{H}^1(\Omega)$ such that

$$-\Delta u = f \quad \text{in } \Omega \subset \mathbb{R}^{\dim}, \tag{4.1}$$

$$u = g_D \quad \text{on } \Gamma_D, \tag{4.2}$$

$$\frac{\partial u}{\partial n} + a_R\, u = g_R \quad \text{on } \Gamma_R, \tag{4.3}$$

where $\Omega \subset \mathbb{R}^{\dim}$ with $\partial\Omega = \Gamma_D \cup \Gamma_R$ and $\Gamma_D \cap \Gamma_R = \emptyset$.

The Laplacian operator Δ can be rewritten according to a $\mathcal{L}$ operator defined in (1.1) and we have

$$-\Delta \stackrel{\mathsf{def}}{=} -\sum_{i=1}^{\dim} \frac{\partial^2}{\partial {x_i}^2} = \mathcal{L}_{\mathbb{I},\mathbf{0},\mathbf{0},0}. \tag{4.4}$$

The conormal derivative $\frac{\partial u}{\partial n_{\mathcal{L}}}$ of this $\mathcal{L}$ operator is given by

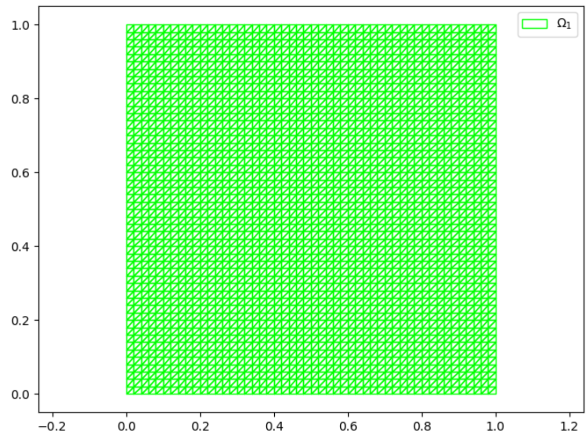
$$\frac{\partial u}{\partial n_{\mathcal{L}}} \stackrel{\mathsf{def}}{=} \langle \mathbb{A}\,\boldsymbol{\nabla}\, u, \boldsymbol{n}\rangle - \langle \boldsymbol{b}u, \boldsymbol{n}\rangle = \frac{\partial u}{\partial n}. \tag{4.5}$$

We now will see how to implement different Poisson's BVP while using the FC-VFEM$\mathbb{P}_1$ toolbox.

4.1.1 2D Poisson BVP with Dirichlet boundary conditions on the unit square

Let Ω be the unit square with the associated mesh obtain from HyperCube function (see section **??** for explanation and Figure **??** for a mesh sample) by the command

```
Th=fc_simesh.siMesh.HyperCube(2,50)
```



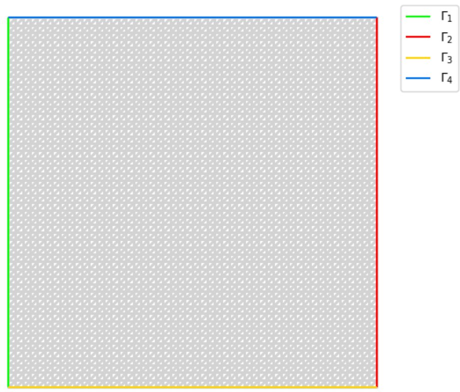


Figure 4.1: 2D hypercube (left) and its boundaries (right)

We choose the problem to have exact solution

$$u_{\text{ex}}(x, y) = \cos(x - y)\sin(x + y) + e^{(-x^2 - y^2)}.$$

So we set $f = -\Delta u_{\text{ex}}$ i.e.

$$f(x, y) = -4\,x^2 e^{(-x^2 - y^2)} - 4\,y^2 e^{(-x^2 - y^2)} + 4\,\cos(x - y)\sin(x + y) + 4\,e^{(-x^2 - y^2)}.$$

On all the 4 boundaries we set a Dirichlet boundary conditions (and so $\Gamma_R = \emptyset$) :

$$u = u_{\text{ex}}, \text{ on } \Gamma_D = \Gamma_1 \cup \Gamma_2 \cup \Gamma_3 \cup \Gamma_4.$$

So this problem can be written as the scalar BVP 5

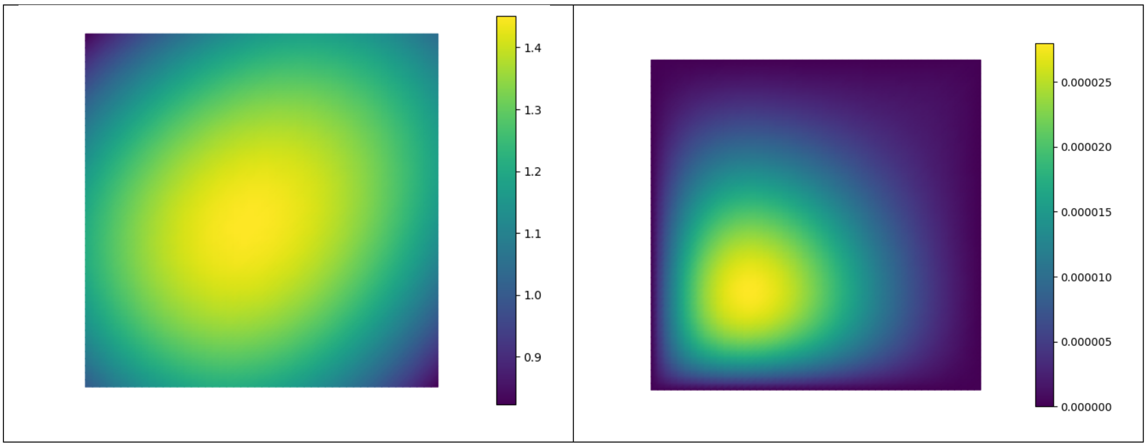
***Scalar* BVP 3 : 2D Poisson BVP with Dirichlet boundary conditions**

Find $u \in \mathrm{H}^1(\Omega)$ such that

$$\mathcal{L}_{\mathbb{I},\mathbf{0},\mathbf{0},0}(u) = f \text{ in } \Omega = [0, 1]^2, \quad (4.6)$$

$$u = u_{\text{ex}} \text{ on } \Gamma_1 \cup \Gamma_2 \cup \Gamma_3 \cup \Gamma_4, \quad (4.7)$$

In Listing 10, we give the complete code to solve this problem with FC-VFEM$\mathbb{P}_1$ toolbox.




```
uex=lambda x,y: np.cos(x-y)*np.sin(x+y)+np.exp(-(x**2+y**2))
f=lambda x,y: -4*x**2*np.exp(-x**2-y**2) - 4*y**2*np.exp(-x**2-y**2) +
    4*np.cos(x-y)*np.sin(x+y) + 4*np.exp(-x**2-y**2)
Th=HyperCube(2,50);
Lop=Loperator(dim=2,A=[[1,None],[None,1]])
pde=PDE(Op=Lop, f=f)
bvp=BVP(Th,pde=pde)
for lab in [1,2,3,4]:
  bvp.setDirichlet( lab, uex)
U=bvp.solve();
```

Listing 4.1: Poisson 2D BVP with Dirichlet boundary conditions : numerical solution (left) and error (right)

4.1.2 2D Poisson BVP with mixed boundary conditions

Let Ω be the unit square with the associated mesh obtain from HyperCube function (see section **??** for explanation and Figure **??** for a mesh sample)

We choose the problem to have exact solution

$$u_{\text{ex}}(x, y) = \cos(2\,x + y)\,.$$

So we set $f = -\Delta u_{\text{ex}}$ i.e.

$$f(x, y) = 5\,\cos(2\,x + y)\,.$$

On boundary labels 1 and 2 we set a Dirichlet boundary conditions :

$$u = u_{\text{ex}}, \text{ on } \Gamma^D = \Gamma_1 \cup \Gamma_2.$$

On boundary label 3, we choose a Robin boundary condition with $a^R(x, y) = x^2 + y^2 + 1$. So we have

$$\frac{\partial u}{\partial n} + a^R u = g^R, \text{ on } \Gamma^R = \Gamma_3$$

with $g^R = \left(x^2 + y^2 + 1\right)\cos(2\,x + y) + \sin(2\,x + y)\,.$

On boundary label 4, we choose a Newmann boundary condition. So we have

$$\frac{\partial u}{\partial n} = g^N, \text{ on } \Gamma^N = \Gamma_4$$

with $g^N = -\sin(2\,x + y)\,.$ this can be also written in the form of a Robin condition with $aR = 0$

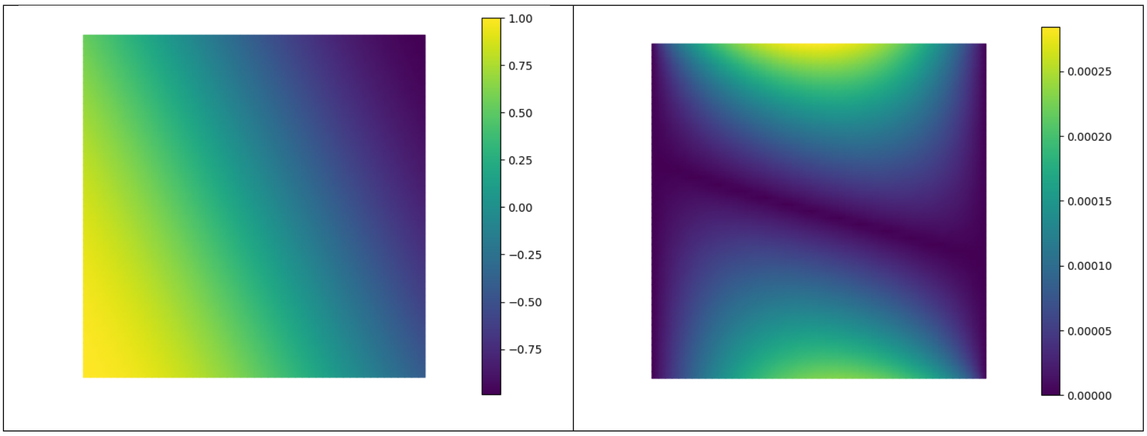
So this problem can be written as the scalar BVP 5

Scalar BVP 4 : 2D Poisson BVP with mixed boundary conditions

Find $u \in \mathrm{H}^1(\Omega)$ such that

$$\begin{aligned} \mathcal{L}_{\mathbb{I},\mathbf{0},\mathbf{0},0}(u) &= f \quad \text{in } \Omega = [0,1]^2, & (4.8)\\ u &= u_{\text{ex}} \quad \text{on } \Gamma_1 \cup \Gamma_2 \cup \Gamma_3 \cup \Gamma_4, & (4.9)\\ \frac{\partial u}{\partial n_{\mathcal{L}}} + a^R u &= g^R \quad \text{on } \Gamma_3, & (4.10)\\ \frac{\partial u}{\partial n_{\mathcal{L}}} &= g^N \quad \text{on } \Gamma_4, & (4.11)\\ & & (4.12) \end{aligned}$$

In Listing 14, we give the complete code to solve this problem with FC-VFEM$\mathbb{P}_1$ toolbox.




```
uex=lambda x,y: np.cos(2*x+y)
f=lambda x,y: 5*np.cos(2*x+y)
gradu=[lambda x,y: -2*np.sin(2*x+y),lambda x,y: -np.sin(2*x+y)]
ar3 = lambda x,y: 1+x**2+y**2
Th=HyperCube(2,50)
Lop=Loperator(dim=2,A=[[1,None],[None,1]])
pde=PDE(Op=Lop,f=f)
bvp=BVP(Th,pde=pde)
bvp.setDirichlet( 1, uex)
bvp.setDirichlet( 2, uex)
bvp.setRobin( 3,lambda x,y: -gradu[1](x,y)+ar3(x,y)*uex(x,y) , ar=ar3)
bvp.setRobin( 4, gradu[1])
U=bvp.solve()
```

Listing 4.2: Poisson 2D BVP with mixed boundary conditions : numerical solution (left) and error (right)

4.1.3 3D Poisson BVP with mixed boundary conditions

Let Ω be the unit cube with the associated mesh obtain from HYPERCUBE function (see section **??** for explanation and Figure **??** for a mesh sample)

We choose the problem to have exact solution

$$u_{\text{ex}}(x,y,y) = \cos(4\,x - 3\,y + 5\,z).$$

So we set $f = -\Delta u_{\text{ex}}$ i.e.

$$f(x,y,z) = 50\,\cos(4\,x - 3\,y + 5\,z).$$

On boundary labels $1,3,5$ we set a Dirichlet boundary conditions :

$$u = u_{\text{ex}}, \ \text{on } \Gamma^D = \Gamma_1 \cup \Gamma_3 \cup \Gamma_5.$$

On boundary label 2, we choose a Robin boundary condition with $a^R(x, y) = 1$. So we have

$$\frac{\partial u}{\partial n} + a^R u = g^R, \text{ on } \Gamma^R = \Gamma_2 \cup \Gamma_4$$

with $g^R(x, y, z) = \cos(4\,x - 3\,y + 5\,z) - 4\,\sin(4\,x - 3\,y + 5\,z)$, on Γ_2 and $g^R(x, y, z) = \cos(4\,x - 3\,y + 5\,z) + 3\,\sin(4\,x - 3\,y + 5\,z)$, on Γ_4.

On boundary label 6, we choose a Newmann boundary condition. So we have

$$\frac{\partial u}{\partial n} = g^N, \text{ on } \Gamma^N = \Gamma_6$$

with $g^N = -5\,\sin(4\,x - 3\,y + 5\,z)$. this can be also written in the form of a Robin condition with $aR = 0$ on Γ_6.

So this problem can be written as the scalar BVP 5

Scalar BVP 5 : 3D Poisson BVP with mixed boundary conditions

Find $u \in \mathrm{H}^1(\Omega)$ such that

$$\mathcal{L}_{\mathbb{I},\mathbf{0},\mathbf{0},0}(u) = f \quad \text{in } \Omega = [0,1]^3, \tag{4.13}$$

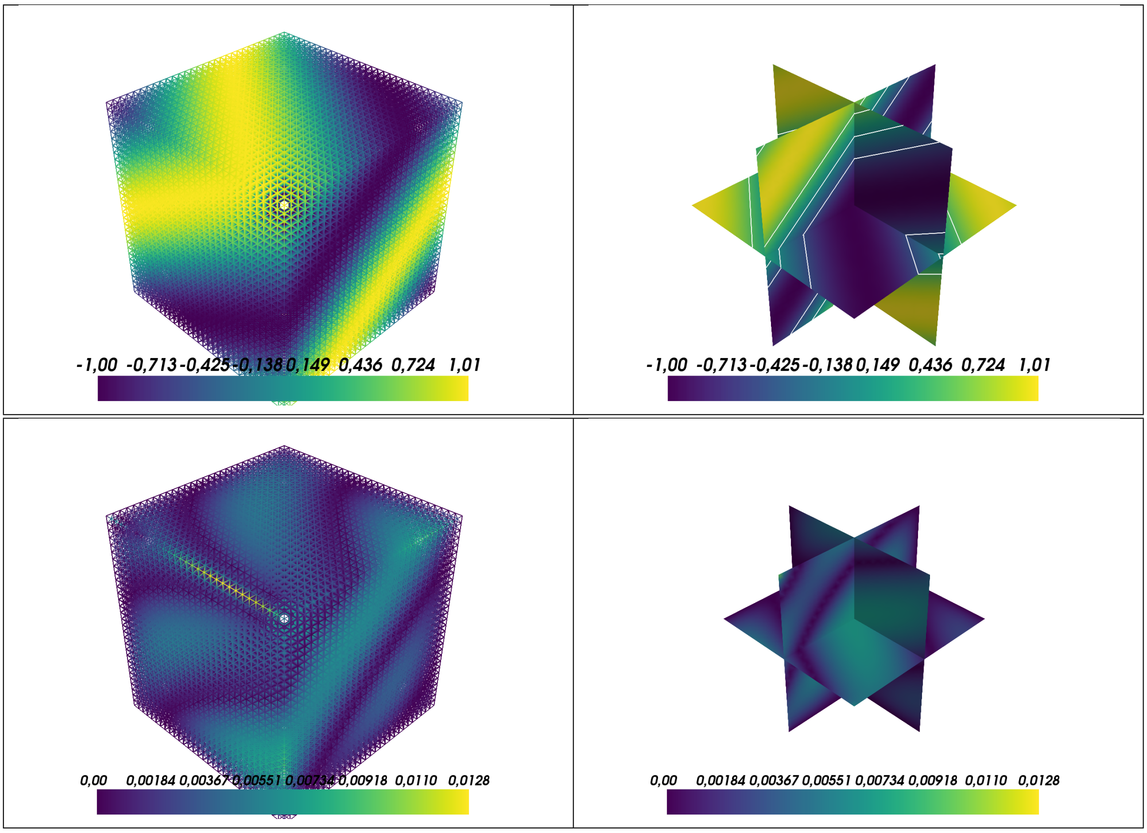
$$u = u_{\text{ex}} \quad \text{on } \Gamma_1 \cup \Gamma_3 \cup \Gamma_5, \tag{4.14}$$

$$\frac{\partial u}{\partial n_{\mathcal{L}}} + a^R u = g^R \quad \text{on } \Gamma_2 \cup \Gamma_4, \tag{4.15}$$

$$\frac{\partial u}{\partial n_{\mathcal{L}}} = g^N \quad \text{on } \Gamma_6, \tag{4.16}$$

(4.17)

In Listing 15, we give the complete code to solve this problem with FC-VFEM$\mathbb{P}_1$ toolbox.




```
uex=lambda x,y,z: np.cos(4*x-3*y+5*z)
f=lambda x,y,z: 50*uex(x,y,z)
gradu=[lambda x,y,z: -4*np.sin(4*x-3*y+5*z), lambda x,y,z:
    3*np.sin(4*x-3*y+5*z), lambda x,y,z: -5*np.sin(4*x-3*y+5*z)]
ar = 1
Th=HyperCube(3,30)
Lop=Loperator(dim=3,d=3,A=[[1,None,None],[None,1,None],[None,None,1]])
pde=PDE(Op=Lop, f=f)
bvp=BVP(Th, pde=pde)
for lab in [1,3,5]:
  bvp.setDirichlet( lab, uex)
bvp.setRobin( 2,lambda x,y,z: gradu[0](x,y,z)+ar*uex(x,y,z) , ar=ar)
bvp.setRobin( 4,lambda x,y,z: gradu[1](x,y,z)+ar*uex(x,y,z) , ar=ar)
bvp.setRobin( 6,gradu[2])
u=bvp.solve();
```

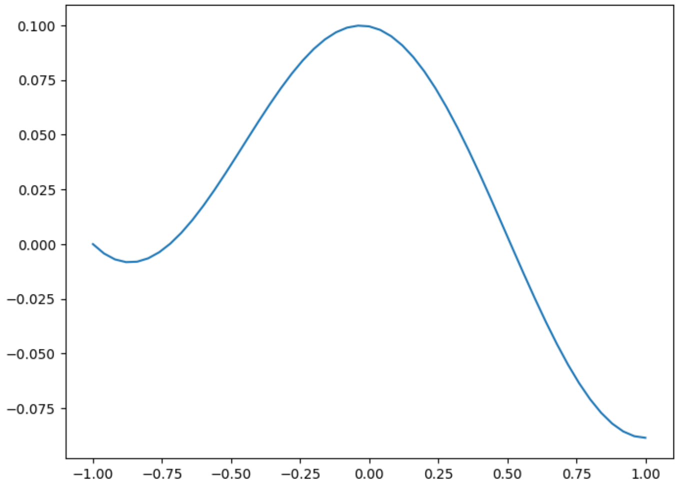
Listing 4.3: 3D Poisson BVP with mixed boundary conditions : numerical solution (upper) and error (bottom)

4.1.4 1D BVP : just for fun

Let Ω be the interval $[a,b]$ we want to solve the following PDE

$$-u''(x) + c(x)u(x) = f(x) \quad \forall x \in]a,b[$$

with the Dirichlet boundary condition $u(a) = 0$ and the homgeneous Neumann boundary condition on b




```
f=lambda x: np.cos(np.pi*x)
c=lambda x: 1+(x-1)**2
a=-1; b=1
Th=HyperCube(1,50,mapping=lambda x: a + (b-a)*x)
Lop=Loperator(dim=1,A=[[1]],a0=c)
pde=PDE(Op=Lop,f=f)
bvp=BVP(Th,pde=pde)
bvp.setDirichlet( 1, 0)
U=bvp.solve()
```

Listing 4.4: 1D BVP with mixed boundary conditions

4.2 Stationary convection-diffusion problem

4.2.1 Stationary convection-diffusion problem in 2D

The 2D problem to solve is the following

Usual BVP 3 : 2D stationary convection-diffusion problem

Find $u \in \mathrm{H}^1(\Omega)$ such that

$$-\operatorname{div}(\alpha \boldsymbol{\nabla} u) + \langle \boldsymbol{V}, \boldsymbol{\nabla} u\rangle + \beta u = f \quad \text{in } \Omega \subset \mathbb{R}^2, \tag{4.18}$$

$$u = 4 \quad \text{on } \Gamma_2, \tag{4.19}$$

$$u = -4 \quad \text{on } \Gamma_4, \tag{4.20}$$

$$u = 0 \quad \text{on } \Gamma_{20} \cup \Gamma_{21}, \tag{4.21}$$

$$\frac{\partial u}{\partial n} = 0 \quad \text{on } \Gamma_1 \cup \Gamma_3 \cup \Gamma_{10} \tag{4.22}$$

where Ω and its boundaries are given in Figure **??**. This problem is well posed if $\alpha(\boldsymbol{x}) > 0$ and $\beta(\boldsymbol{x}) \geqslant 0$.
We choose α, $\boldsymbol{V}$, β and f in Ω as :

$$\begin{aligned}
\alpha(\boldsymbol{x}) &= 0.1 + (x_1 - 0.5)^2, \\
\boldsymbol{V}(\boldsymbol{x}) &= (-10x_2, 10x_1)^t, \\
\beta(\boldsymbol{x}) &= 0.01, \\
f(\boldsymbol{x}) &= -200\exp(-10((x_1 - 0.75)^2 + x_2^2)).
\end{aligned}$$

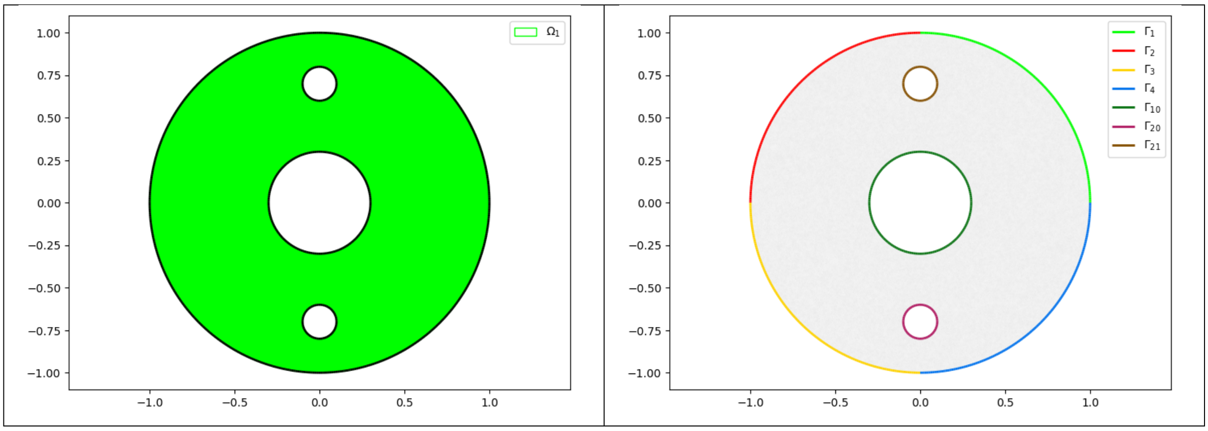


Figure 4.2: 2D stationary convection-diffusion BVP : mesh (left) and boundaries (right) representations by using **matplotlib** package

The problem (4.18)-(4.22) can be equivalently expressed as the scalar BVP (1.2)-(1.4) :

Scalar BVP 6 : 2D stationary convection-diffusion problem

Find $u \in \mathrm{H}^1(\Omega)$ such that

$$\begin{aligned} \mathcal{L}(u) &= f & \text{in } \Omega, \\ u &= g^D & \text{on } \Gamma^D, \\ \frac{\partial u}{\partial n_{\mathcal{L}}} + a^R u &= g^R & \text{on } \Gamma^R. \end{aligned}$$

where

- $\mathcal{L} := \mathcal{L}_{\alpha\mathbb{I},\mathbf{0},\boldsymbol{V},\beta}$, and then the conormal derivative of u is given by

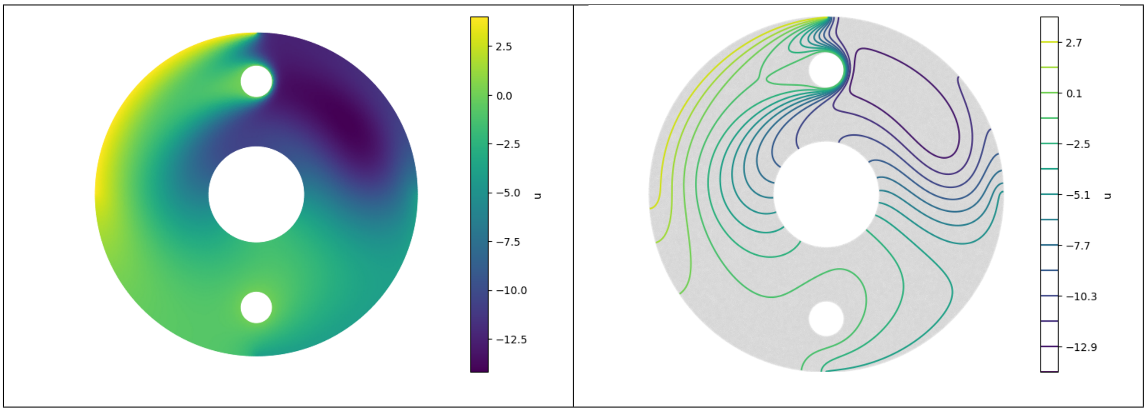
$$\frac{\partial u}{\partial n_{\mathcal{L}}} := \langle \mathbb{A}\boldsymbol{\nabla} u, \boldsymbol{n}\rangle - \langle \boldsymbol{b}u, \boldsymbol{n}\rangle = \alpha\frac{\partial u}{\partial n}.$$

- $\Gamma^D = \Gamma_2 \cup \Gamma_4 \cup \Gamma_{20} \cup \Gamma_{21}$ and $\Gamma^R = \Gamma_1 \cup \Gamma_3 \cup \Gamma_{10}$
- $g^D := 4$ on Γ_2, and $g^D := -4$ on Γ_4 and $g^D := 0$ on $\Gamma_{20} \cup \Gamma_{21}$
- $a^R = g^R := 0$ on Γ^R.

The algorithm using the toolbox for solving (4.18)-(4.22) is the following:

Algorithm 1 Stationary convection-diffusion problem in 2D

$\mathcal{T}_h \leftarrow$ siMesh(...) ▷ Get mesh
$\alpha \leftarrow (x,y) \longmapsto 0.1 + (y-0.5)(y-0.5)$
$\beta \leftarrow 0.01$
$f \leftarrow (x,y) \longmapsto -200e^{-10((x-0.75)^2+y^2)}$
Lop ← Loperator$\left(2, 2, \begin{pmatrix} \alpha & 0 \\ 0 & \alpha \end{pmatrix}, \mathbf{0}, \begin{pmatrix} -10y \\ 10x \end{pmatrix}, \beta\right)$
pde ← PDEelt(Lop, f)
bvp ← BVP($\mathcal{T}_h$, pde)
bvp.setDirichlet(2, 4.0) ▷ Set 'Dirichlet' condition on Γ_2
bvp.setDirichlet(4, −4.0) ▷ Set 'Dirichlet' condition on Γ_4
bvp.setDirichlet(20, 0.0) ▷ Set 'Dirichlet' condition on Γ_{20}
bvp.setDirichlet(21, 0.0) ▷ Set 'Dirichlet' condition on Γ_{21}
$\boldsymbol{u} \leftarrow$ bvp.solve()




```
geofile='disk3holes'
af=lambda x,y: 0.1+(y-0.5)*(y-0.5)
Vx=lambda x,y: -10*y
Vy=lambda x,y: 10*x
b=0.01;g2=4;g4=-4;
f=lambda x,y: -200.0*exp(-((x-0.75)**2+y**2)/(0.1))

meshfile=fc_oogmsh.gmsh.buildmesh2d(geofile,N,force=True,verbose=0)

pde=PDE(Op=Loperator(dim=2,d=2,A=[[af,None],[None,af]],c=[Vx,Vy],a0=b),f=f)
bvp=BVP(Th,pde=pde)
bvp.setDirichlet(2,g2)
bvp.setDirichlet(4,g4)
bvp.setDirichlet(20,0)
```

Listing 4.5: Setting the 2D stationary convection-diffusion BVP and representation of the numerical solution by using **matplotlib** package

The numerical solution for a given mesh is shown on figures of Listing 4.5

4.2.2 Stationary convection-diffusion problem in 3D

Let $A = (x_A, y_A) \in \mathbb{R}^2$ and $\mathcal{C}^r_A([z_{min}, z_{max}])$ be the right circular cylinder along $z-$axis ($z \in [z_{min}, z_{max}]$) with bases the circles of radius r and center (x_A, y_A, z_{min}) and (x_A, y_A, z_{max}).

Let Ω be the cylinder defined by

$$\Omega = \mathcal{C}^1_{(0,0)}([0,3]) \backslash \{\mathcal{C}^{0.3}_{(0,0)}([0,3]) \cup \mathcal{C}^{0.1}_{(0,-0.7)}([0,3]) \cup \mathcal{C}^{0.1}_{(0,0.7)}([0,3])\}.$$

We respectively denote by Γ_{1000} and Γ_{1001} the $z = 0$ and $z = 3$ bases of Ω.

Γ_1, Γ_{10}, Γ_{20} and Γ_{21} are respectively the curved surfaces of cylinders $\mathcal{C}^1_{(0,0)}([0,3])$, $\mathcal{C}^{0.3}_{(0,0)}([0,3])$, $\mathcal{C}^{0.1}_{(0,-0.7)}([0,3])$ and $\mathcal{C}^{0.1}_{(0,0.7)}([0,3])$.

The domain Ω and its boundaries are represented in Figure 4.3.

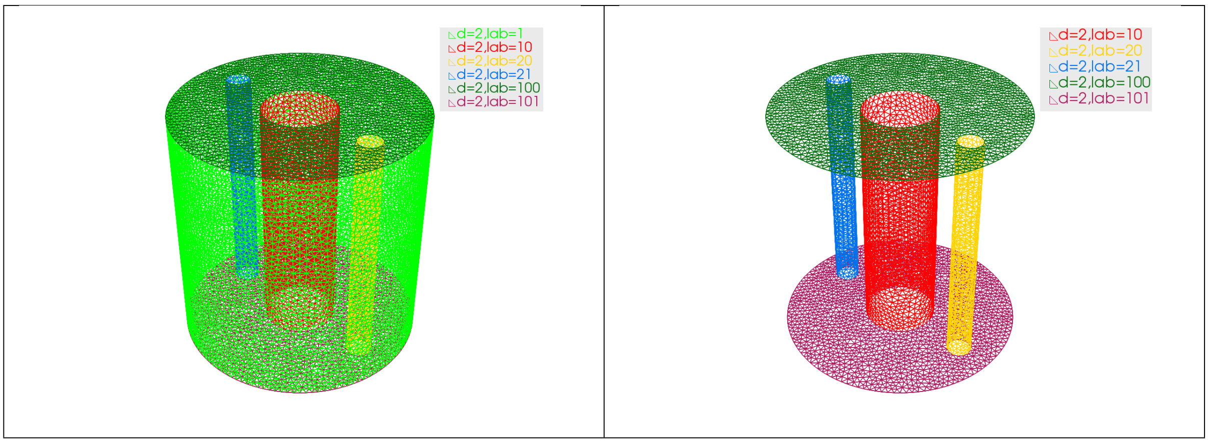


Figure 4.3: 3D stationary convection-diffusion BVP : all boundaries (left) and boundaries without Γ_1 (right) representations by using **Mayavi** package

The 3D problem to solve is the following

***Usual* BVP 4 :**

3D problem : Stationary convection-diffusion Find $u \in \mathrm{H}^2(\Omega)$ such that

$$-\operatorname{div}(\alpha \boldsymbol{\nabla} u) + \langle \boldsymbol{V}, \boldsymbol{\nabla} u\rangle + \beta u = f \quad \text{in } \Omega \subset \mathbb{R}^3, \tag{4.23}$$

$$\alpha \frac{\partial u}{\partial n} + a_{20} u = g_{20} \quad \text{on } \Gamma_{20}, \tag{4.24}$$

$$\alpha \frac{\partial u}{\partial n} + a_{21} u = g_{21} \quad \text{on } \Gamma_{21}, \tag{4.25}$$

$$\frac{\partial u}{\partial n} = 0 \quad \text{on } \Gamma^N \tag{4.26}$$

where $\Gamma^N = \Gamma_1 \cup \Gamma_{10} \cup \Gamma_{1000} \cup \Gamma_{1001}$. This problem is well posed if $\alpha(\boldsymbol{x}) > 0$ and $\beta(\boldsymbol{x}) \geqslant 0$. We choose $a_{20} = a_{21} = 1$, $g_{21} = -g_{20} = 0.05$ $\beta = 0.01$ and :

$$\begin{aligned}
\alpha(\boldsymbol{x}) &= 0.7 + \boldsymbol{x}_3/10, \\
\boldsymbol{V}(\boldsymbol{x}) &= (-10x_2, 10x_1, 10x_3)^t, \\
f(\boldsymbol{x}) &= -800 \exp(-10((x_1 - 0.65)^2 + x_2^2 + (x_3 - 0.5)^2)) \\
&\quad +800 \exp(-10((x_1 + 0.65)^2 + x_2^2 + (x_3 - 0.5)^2)).
\end{aligned}$$

The problem (4.23)-(4.26) can be equivalently expressed as the scalar BVP (1.2)-(1.4) :

***Scalar* BVP 7 :**

3D stationary convection-diffusion problem as a *scalar* BVP Find $u \in \mathrm{H}^2(\Omega)$ such that

$$\begin{aligned}
\mathcal{L}(u) &= f && \text{in } \Omega, \\
\frac{\partial u}{\partial n_{\mathcal{L}}} + a^R u &= g^R && \text{on } \Gamma^R.
\end{aligned}$$

where

- $\mathcal{L} := \mathcal{L}_{\alpha\mathbb{I}, \mathbf{0}, \boldsymbol{V}, \beta}$, and then the conormal derivative of u is given by

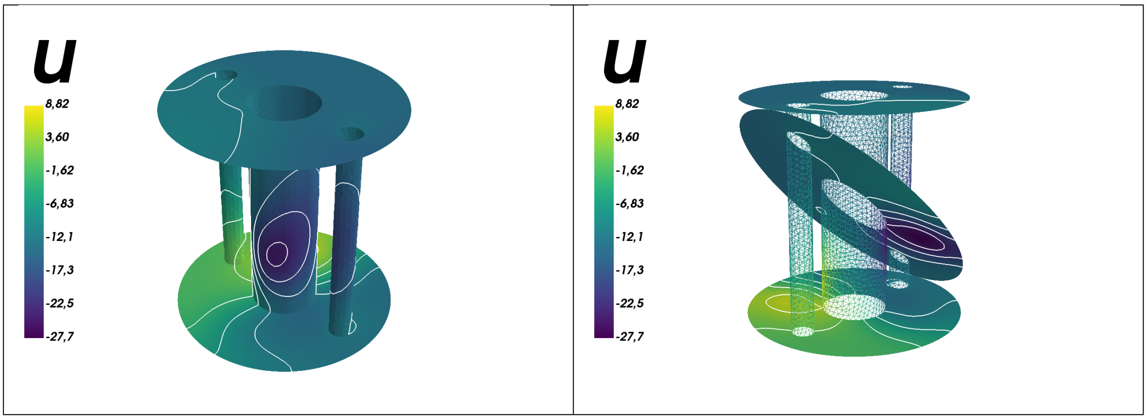
$$\frac{\partial u}{\partial n_{\mathcal{L}}} := \langle \mathbb{A} \boldsymbol{\nabla} u, \boldsymbol{n}\rangle - \langle \boldsymbol{b} u, \boldsymbol{n}\rangle = \alpha \frac{\partial u}{\partial n}.$$

- $\Gamma^R = \Gamma_1 \cup \Gamma_{10} \cup \Gamma_{20} \cup \Gamma_{21} \cup \Gamma_{1000} \cup \Gamma_{1001}$ (and $\Gamma^D = \emptyset$)
-

$$a^R = \begin{cases} 0 & \text{on } \Gamma_1 \cup \Gamma_{10} \cup \Gamma_{1000} \cup \Gamma_{1001} \\ 1 & \text{on } \Gamma_{20} \cup \Gamma_{21} \end{cases}$$

$$g^R = \begin{cases} 0 & \text{on } \Gamma_1 \cup \Gamma_{10} \cup \Gamma_{1000} \cup \Gamma_{1001} \\ 0.05 & \text{on } \Gamma_{21}, \\ -0.05 & \text{on } \Gamma_{20} \end{cases}$$

We give respectively in Listing 4.6 the corresponding Python codes and the numerical solution for a more refined mesh.




```
 verbose=10
geofile='cylinder3holes'
af=lambda x,y,z: 0.7+z/10
Vx=lambda x,y,z: -10*y;Vy=lambda x,y,z: 10*x;Vz=lambda x,y,z: 10*z
f=lambda x,y,z: -800.0*np.exp(-10*((x-0.65)**2+y*y+(z-0.5)**2))+\
                800.0*np.exp(-10*((x+0.65)**2+y*y+(z-0.5)**2))
b=0.01
meshfile=fc_oogmsh.gmsh.buildmesh3d(geodir+os.sep+geofile+'.geo',N, force=True,verbose=0,meshdir=meshdir)
Th=siMesh(meshfile)
pde=PDE(Op=Loperator(dim=3,d=3,A=[[af,None,None],[None,af,None],[None,None,af]], c=[Vx,Vy,Vy],a0=b),f=f)
bvp=BVP(Th,pde=pde)
bvp.setRobin(20,+0.05,ar=1)
bvp.setRobin(21,-0.05,ar=1)
```

Listing 4.6: Setting the 3D stationary convection-diffusion BVP and representation of the numerical solution by using **Mayavi** package

4.3 2D electrostatic BVPs

In this sample, we shall discuss electrostatic solutions for current flow in resistive media. Consider a region Ω of contiguous solid and/or liquid conductors. Let $\boldsymbol{j}$ be the current density in A/m^2. It's satisfy

$$\operatorname{div} \boldsymbol{j} = 0, \quad \text{in } \Omega. \tag{4.27}$$

$$\boldsymbol{j} = \sigma \boldsymbol{E}, \quad \text{in } \Omega. \tag{4.28}$$

where σ is the local electrical conductivity and $\boldsymbol{E}$ the local electric field.

The electric field can be written as a gradient of a scalar potential

$$\boldsymbol{E} = -\boldsymbol{\nabla}\varphi, \quad \text{in } \Omega. \tag{4.29}$$

Combining all these equations leads to Laplace's equation

$$\operatorname{div}(\sigma \boldsymbol{\nabla}\varphi) = 0 \tag{4.30}$$

In the resistive model, a good conductor has high value of σ and a good insulator has $0 < \sigma \ll 1$. This table shows the resistivity ρ and the conductivity σ of various materials at $20°C$:

Material	$\rho(\Omega.m)$ at $20°C$	$\sigma(S/m)$ at $20°C$
Carbon (graphene)	1.00×10^{-8}	1.00×10^{8}
Gold	2.44×10^{-8}	4.10×10^{8}
Aluminium	2.82×10^{-8}	3.50×10^{7}
Zinc	5.90×10^{-8}	1.69×10^{7}
Drinking water	2.00×10^{1} to 2.00×10^{3}	5.00×10^{-4} to 5.00×10^{-2}
Silicon	6.40×10^{2}	1.56×10^{-3}
Glass	1.00×10^{11} to 1.00×10^{15}	10^{-15} to 10^{-11}
Air	1.30×10^{16} to 3.30×10^{16}	3×10^{-15} to 8×10^{-15}

As example, we use the mesh obtain with `gmsh` from `square4holes6dom.geo` file represented in Figure 4.4

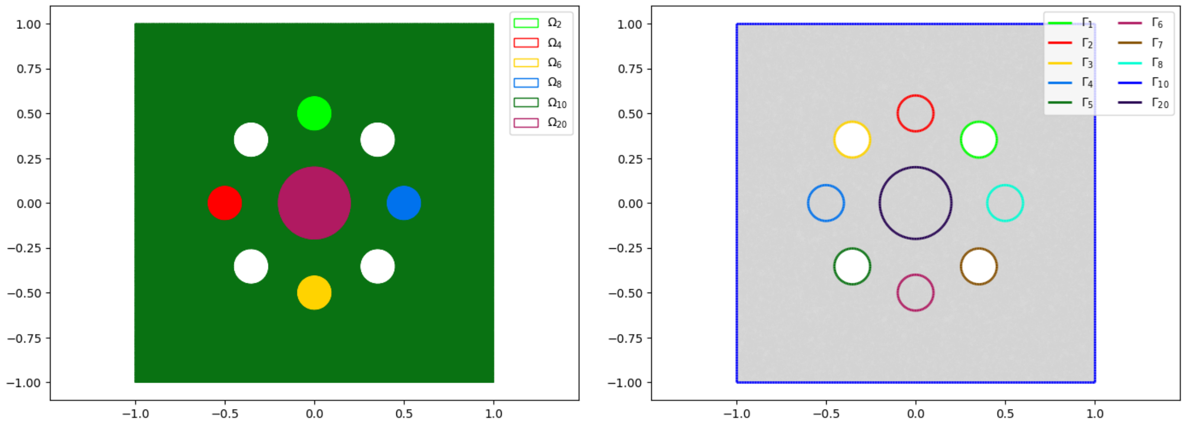


Figure 4.4: Mesh from `square4holes6dom.geo`, domains representation (left) and boundaries (right)

We have two resistive medias

$$\Omega_a = \Omega_{10} \quad \text{and} \quad \Omega_b = \Omega_{20} \cup \Omega_2 \cup \Omega_4 \cup \Omega_6 \cup \Omega_8.$$

In Ω_a and Ω_b the local electrical conductivity are respectively given by

$$\sigma = \begin{cases} \sigma_a & = & 10^4, & \text{in } \Omega_a \\ \sigma_b & = & 10^{-4} & \text{in } \Omega_a \end{cases}$$

We solve the following BVP

Usual BVP 5 : 2D electrostatic problem

Find $\varphi \in \mathrm{H}^1(\Omega)$ such that

$$\operatorname{div}(\sigma \boldsymbol{\nabla} \varphi) = 0 \qquad \text{in } \Omega, \qquad (4.31)$$

$$\varphi = 0 \qquad \text{on } \Gamma_3 \cup \Gamma_7, \qquad (4.32)$$

$$\varphi = 12 \qquad \text{on } \Gamma_1 \cup \Gamma_5, \qquad (4.33)$$

$$\sigma \frac{\partial \varphi}{\partial n} = 0 \qquad \text{on } \Gamma_{10}. \qquad (4.34)$$

The problem (4.31)-(4.34) can be equivalently expressed as the scalar BVP (1.2)-(1.4) :

Scalar BVP 8 : 2D electrostatic problem

Find $\varphi \in \mathrm{H}^1(\Omega)$ such that

$$\mathcal{L}(\varphi) = 0 \qquad \text{in } \Omega,$$

$$\varphi = g^D \qquad \text{on } \Gamma^D,$$

$$\frac{\partial \varphi}{\partial n_{\mathcal{L}}} + a^R \varphi = g^R \qquad \text{on } \Gamma^R.$$

where

- $\mathcal{L} := \mathcal{L}_{\sigma\mathbb{I},\mathbf{0},\boldsymbol{V},\beta}$, and then the conormal derivative of φ is given by

$$\frac{\partial \varphi}{\partial n_{\mathcal{L}}} := \langle \mathbb{A} \boldsymbol{\nabla} \varphi, \boldsymbol{n} \rangle - \langle \boldsymbol{b} \varphi, \boldsymbol{n} \rangle = \sigma \frac{\partial \varphi}{\partial n}.$$

- $\Gamma^D = \Gamma_1 \cup \Gamma_3 \cup \Gamma_5 \cup \Gamma_7$ and $\Gamma^R = \Gamma_{10}$. The other borders should not be used to specify boundary conditions: they do not intervene in the variational formulation and in the physical problem!
- $g^D := 0$ on $\Gamma_3 \cup \Gamma_7$, and $g^D := 12$ on $\Gamma_1 \cup \Gamma_5$.
- $a^R = g^R := 0$ on Γ^R.

To write this problem properly with FC-VFEM$\mathbb{P}_1$ toolbox, we split (4.31) in two parts

$$\begin{aligned} \operatorname{div}(\sigma_a \boldsymbol{\nabla} \varphi) &= 0 && \text{in } \Omega_a \\ \operatorname{div}(\sigma_b \boldsymbol{\nabla} \varphi) &= 0 && \text{in } \Omega_b \end{aligned}$$

and we set these PDEs on each domains. This is done in Python Listing 4.7.

Listing 4.7: **Setting the 2D electrostatic BVP, Python code**

```
    if verbose>=10:
      return
  meshfile=fc_oogmsh.gmsh.buildmesh2d(geodir+os.sep+geofile+'.geo',N,force=True)
  Th=siMesh(meshfile)
  if verbose>=2:
    print('     Mesh sizes : nq=%d, nme=%d, 
        h=%.3e'%(Th.nq,Th.get_nme(),Th.get_h()))
    print('2. Setting 2D Electrostatic BVP')

  pde=PDE(Op=Loperator(dim=2,A=[[sigma2,None],[None,sigma2]]))
  bvp=BVP(Th,pde=pde)
```

We show in Figures 4.5 and 4.6 respectively the potential φ and the norm of the electric field $\boldsymbol{E}$.

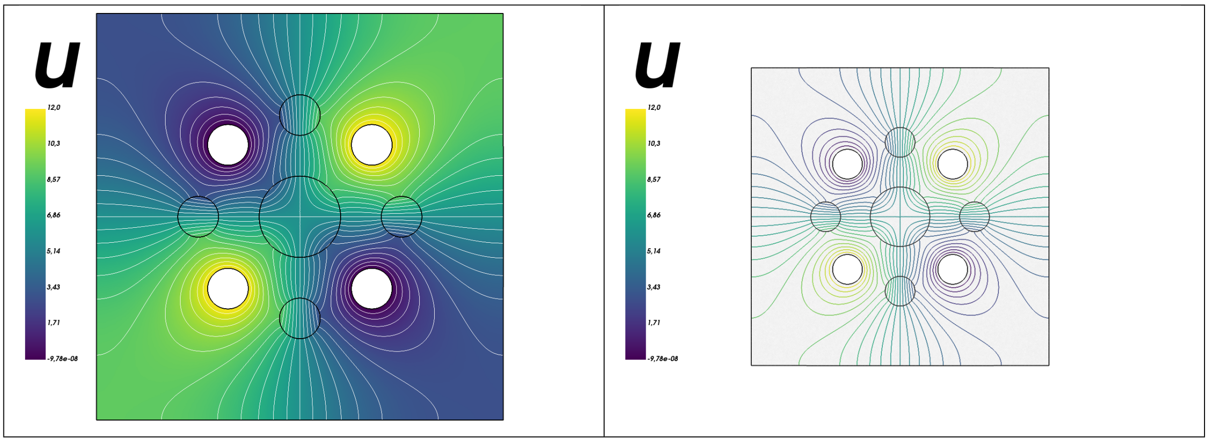


Figure 4.5: Test 1, potential φ

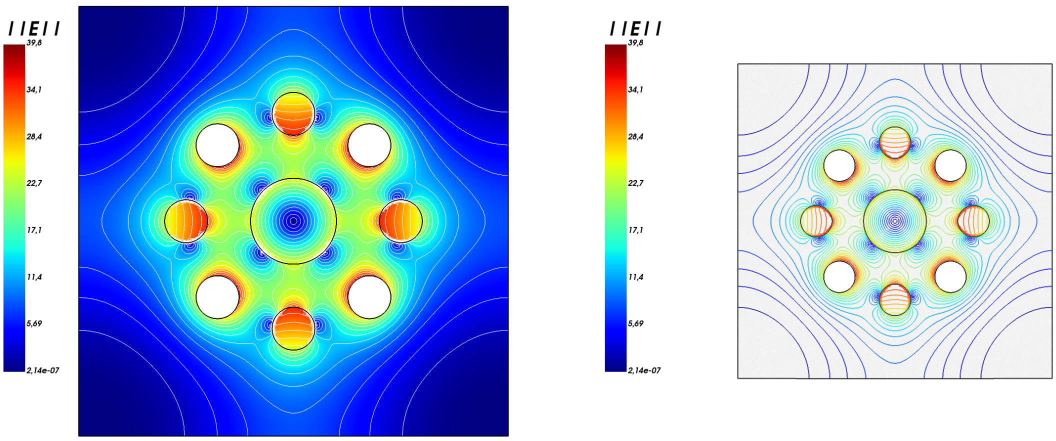


Figure 4.6: Test 1, norm of the electrical field $\boldsymbol{E}$

Chapter 5

Vector boundary value problems

5.1 Elasticity problem

5.1.1 General case ($d = 2, 3$)

We consider here Hooke's law in linear elasticity, under small strain hypothesis (see for example [?]).

For a sufficiently regular vector field $\boldsymbol{u} = (u_1, \dots, u_d) : \Omega \to \mathbb{R}^d$, we define the linearized strain tensor $\underline{\boldsymbol{\epsilon}}$ by

$$\underline{\boldsymbol{\epsilon}}(\boldsymbol{u}) = \frac{1}{2}\left(\boldsymbol{\nabla}(\boldsymbol{u}) + \boldsymbol{\nabla}^t(\boldsymbol{u})\right).$$

We set $\underline{\boldsymbol{\epsilon}} = (\epsilon_{11}, \epsilon_{22}, 2\epsilon_{12})^t$ in 2d and $\underline{\boldsymbol{\epsilon}} = (\epsilon_{11}, \epsilon_{22}, \epsilon_{33}, 2\epsilon_{12}, 2\epsilon_{23}, 2\epsilon_{13})^t$ in 3d, with $\epsilon_{ij}(\boldsymbol{u}) = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right)$. Then the Hooke's law writes

$$\underline{\boldsymbol{\sigma}} = \mathbb{C}\underline{\boldsymbol{\epsilon}},$$

where $\underline{\boldsymbol{\sigma}}$ is the elastic stress tensor and $\mathbb{C}$ the elasticity tensor.

The material is supposed to be isotropic. Thus the elasticity tensor $\mathbb{C}$ is only defined by the Lamé parameters λ and μ, which satisfy $\lambda + \mu > 0$. We also set $\gamma = 2\,\mu + \lambda$. For $d = 2$ or $d = 3$, $\mathbb{C}$ is given by

$$\mathbb{C} = \begin{pmatrix} \lambda \mathbb{1}_2 + 2\mu \mathbb{I}_2 & 0 \\ 0 & \mu \end{pmatrix}_{3\times 3} \quad \text{or} \quad \mathbb{C} = \begin{pmatrix} \lambda \mathbb{1}_3 + 2\mu \mathbb{I}_3 & 0 \\ 0 & \mu \mathbb{I}_3 \end{pmatrix}_{6\times 6},$$

respectively, where $\mathbb{1}_d$ is a d-by-d matrix of ones, and $\mathbb{I}_d$ the d-by-d identity matrix.

For dimension $d = 2$ or $d = 3$, we have:

$$\boldsymbol{\sigma}_{\alpha\beta}(\boldsymbol{u}) = 2\,\mu\,\boldsymbol{\epsilon}_{\alpha\beta}(\boldsymbol{u}) + \lambda\,\mathrm{tr}(\boldsymbol{\epsilon}(\boldsymbol{u}))\delta_{\alpha\beta} \quad \forall \alpha, \beta \in [\![1, d]\!]$$

The problem to solve is the following

Usual vector BVP 2 : Elasticity problem

Find $\boldsymbol{u} = \mathrm{H}^2(\Omega)^d$ such that

$$\begin{aligned} -\operatorname{div}(\boldsymbol{\sigma}(\boldsymbol{u})) &= \boldsymbol{f}, \quad \text{in } \Omega \subset \mathbb{R}^d, && (5.1)\\ \boldsymbol{\sigma}(\boldsymbol{u}).\boldsymbol{n} &= \boldsymbol{0} \quad \text{on } \Gamma^R, && (5.2)\\ \boldsymbol{u} &= \boldsymbol{0} \quad \text{on } \Gamma^D. && (5.3) \end{aligned}$$

Now, with the following lemma, we obtain that this problem can be rewritten as the vector BVP defined by (1.14) to (1.16).

Lemme 5.1

Let $\mathcal{H}$ be the d-by-d matrix of the second order linear differential operators defined in (1.10) where $\mathcal{H}_{\alpha,\beta} = \mathcal{L}_{\mathbb{A}^{\alpha,\beta},\mathbf{0},\mathbf{0},0}$, $\forall(\alpha,\beta) \in [\![1,d]\!]^2$, with

$$(\mathbb{A}^{\alpha,\beta})_{k,l} = \mu\delta_{\alpha\beta}\delta_{kl} + \mu\delta_{k\beta}\delta_{l\alpha} + \lambda\delta_{k\alpha}\delta_{l\beta},\ \forall(k,l) \in [\![1,d]\!]^2. \tag{5.4}$$

then

$$\mathcal{H}(\boldsymbol{u}) = -\operatorname{div}\boldsymbol{\sigma}(\boldsymbol{u}) \tag{5.5}$$

and, $\forall\alpha \in [\![1,d]\!]$,

$$\frac{\partial \boldsymbol{u}}{\partial n_{\mathcal{H}_\alpha}} = (\boldsymbol{\sigma}(\boldsymbol{u}).\boldsymbol{n})_\alpha. \tag{5.6}$$

The proof is given in appendix ??. So we obtain

Vector BVP 3 : Elasticity problem with $\mathcal{H}$ operator in dimension $d = 2$ or $d = 3$

Let $\mathcal{H}$ be the d-by-d matrix of the second order linear differential operators defined in (1.10) where $\forall(\alpha,\beta) \in [\![1,d]\!]^2$, $\mathcal{H}_{\alpha,\beta} = \mathcal{L}_{\mathbb{A}^{\alpha,\beta},\mathbf{0},\mathbf{0},0}$, with

- for $d = 2$,
$$\mathbb{A}^{1,1} = \begin{pmatrix}\gamma & 0\\ 0 & \mu\end{pmatrix},\ \mathbb{A}^{1,2} = \begin{pmatrix}0 & \lambda\\ \mu & 0\end{pmatrix},\ \mathbb{A}^{2,1} = \begin{pmatrix}0 & \mu\\ \lambda & 0\end{pmatrix},\ \mathbb{A}^{2,2} = \begin{pmatrix}\mu & 0\\ 0 & \gamma\end{pmatrix}$$

- for $d = 3$,
$$\begin{aligned}
\mathbb{A}^{1,1} &= \begin{pmatrix}\gamma & 0 & 0\\ 0 & \mu & 0\\ 0 & 0 & \mu\end{pmatrix}, & \mathbb{A}^{1,2} &= \begin{pmatrix}0 & \lambda & 0\\ \mu & 0 & 0\\ 0 & 0 & 0\end{pmatrix}, & \mathbb{A}^{1,3} &= \begin{pmatrix}0 & 0 & \lambda\\ 0 & 0 & 0\\ \mu & 0 & 0\end{pmatrix}\\
\mathbb{A}^{2,1} &= \begin{pmatrix}0 & \mu & 0\\ \lambda & 0 & 0\\ 0 & 0 & 0\end{pmatrix}, & \mathbb{A}^{2,2} &= \begin{pmatrix}\mu & 0 & 0\\ 0 & \gamma & 0\\ 0 & 0 & \mu\end{pmatrix}, & \mathbb{A}^{2,3} &= \begin{pmatrix}0 & 0 & 0\\ 0 & 0 & \lambda\\ 0 & \mu & 0\end{pmatrix},\\
\mathbb{A}^{3,1} &= \begin{pmatrix}0 & 0 & \mu\\ 0 & 0 & 0\\ \lambda & 0 & 0\end{pmatrix}, & \mathbb{A}^{3,2} &= \begin{pmatrix}0 & 0 & 0\\ 0 & 0 & \mu\\ 0 & \lambda & 0\end{pmatrix}, & \mathbb{A}^{3,3} &= \begin{pmatrix}\mu & 0 & 0\\ 0 & \mu & 0\\ 0 & 0 & \gamma\end{pmatrix}.
\end{aligned}$$

The elasticity problem (5.1) to (5.3) can be rewritten as :
Find $\boldsymbol{u} = (\boldsymbol{u}_1, \dots, \boldsymbol{u}_d) \in (\mathrm{H}^2(\Omega))^d$ such that

$$\mathcal{H}(\boldsymbol{u}) = \boldsymbol{f}, \quad \text{in } \Omega, \tag{5.7}$$

$$\frac{\partial \boldsymbol{u}}{\partial n_{\mathcal{H}_\alpha}} = 0, \quad \text{on } \Gamma_\alpha^R = \Gamma^R,\ \forall\alpha \in [\![1,d]\!] \tag{5.8}$$

$$\boldsymbol{u}_\alpha = 0, \quad \text{on } \Gamma_\alpha^D = \Gamma^D,\ \forall\alpha \in [\![1,d]\!]. \tag{5.9}$$

5.1.2 2D example

For example, in 2d, we want to solve the elasticity problem (5.1) to (5.3) where Ω and its boundaries are given in Figure 5.1.

The material's properties are given by Young's modulus E and Poisson's coefficient ν. As we use plane strain hypothesis, Lame's coefficients verify

$$\mu = \frac{E}{2\,(1+\nu)},\quad \lambda = \frac{E\,\nu}{(1+\nu)\,(1-2\,\nu)},\quad \gamma = 2\mu + \lambda$$

The material is rubber so that $E = 21.10^5$Pa and $\nu = 0.45$. We also have $\boldsymbol{f} = \boldsymbol{x} \mapsto (0,-1)^t$ and we choose $\Gamma^R = \Gamma^1 \cup \Gamma^2 \cup \Gamma^3$, $\Gamma^D = \Gamma^4$.

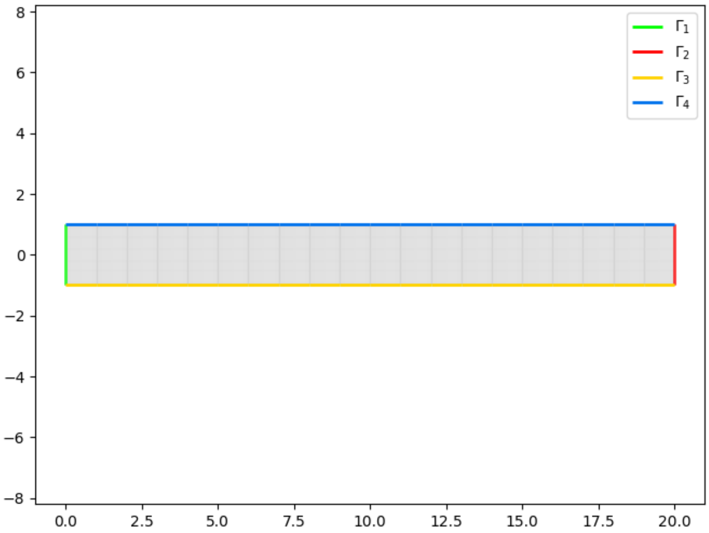


Figure 5.1: Domain and boundaries

Listing 5.1: **2D elasticity, Python code**

```
mu= E/(2*(1+nu))
lam = E*nu/((1+nu)*(1-2*nu))
gam=lam+2*mu
uex=[lambda x,y: cos(2*x+y), lambda x,y: sin(x-3*y)]
f=[lambda x,y: 4*lam*cos(2*x + y) + 9*mu*cos(2*x + y) + 3*lam*sin(-x + 3*y) +
    3*mu*sin(-x + 3*y) ,
    lambda x,y: 2*lam*cos(2*x + y) + 2*mu*cos(2*x + y) - 9*lam*sin(-x + 3*y) -
        19*mu*sin(-x + 3*y)]
g2=[lambda x,y: -3*lam*cos(-x + 3*y) - 2*lam*sin(2*x + y) - 4*mu*sin(2*x + y),
    lambda x,y: mu*(cos(-x + 3*y) - sin(2*x + y))]
print('-----------------------------------------------------')
Th=HyperCube(2,[2*N,20*N],mapping=mapping)
print('     Mesh sizes : nq=%d, nme=%d, h=%.3e'%(Th.nq,Th.get_nme(),Th.get_h()))
Hop.H[0][0]=Loperator(d=2,A=[[gam,None],[None,mu]])
Hop.H[0][1]=Loperator(d=2,A=[[None,lam],[mu,None]])
Hop.H[1][0]=Loperator(d=2,A=[[None,mu],[lam,None]])
Hop.H[1][1]=Loperator(d=2,A=[[mu,None],[None,gam]])
pde=PDE(Op=Hop,f=[0,-1])
bvp=BVP(Th,pde=pde)
bvp.setDirichlet(1,[0,0])
```

One can also use the Python function StiffElasHoperator from fc_vfemp1.operators module to build the elasticity operator :

Hop=StiffElasHoperator(2,lam,mu)

For a given mesh, its displacement scaled by a factor 50 is shown on Figure 5.2

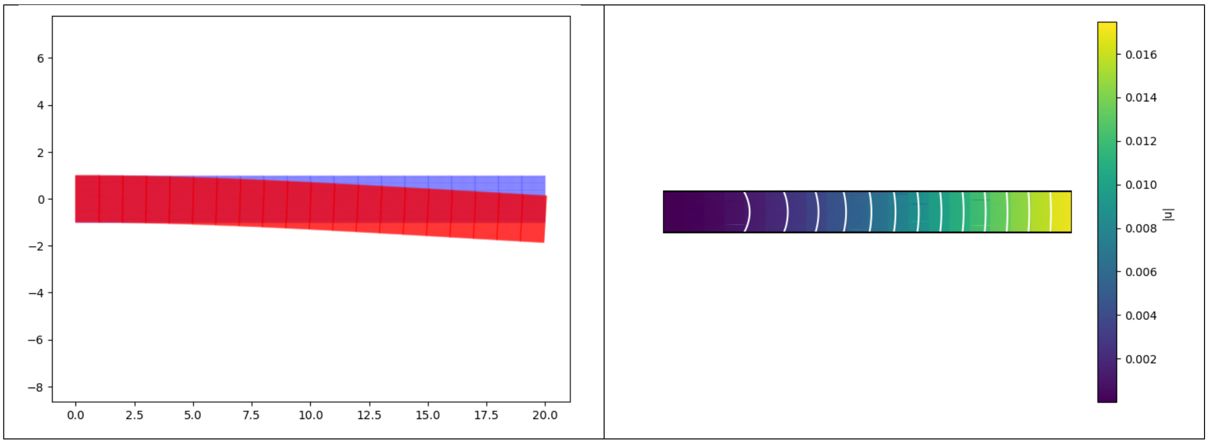


Figure 5.2: 2D elasticity problem: mesh displacement scaled by a factor 50 (left) and norm of the displacement (right)

5.1.3 3D example

Let $\Omega = [0,5]\times[0,1]\times[0,1] \subset \mathbb{R}^3$. The boundary of Ω is made of six faces and each one has a unique label : 1 to 6 respectively for faces $x_1 = 0$, $x_1 = 5$, $x_2 = 0$, $x_2 = 1$, $x_3 = 0$ and $x_3 = 1$. We represent them in Figure 5.3.

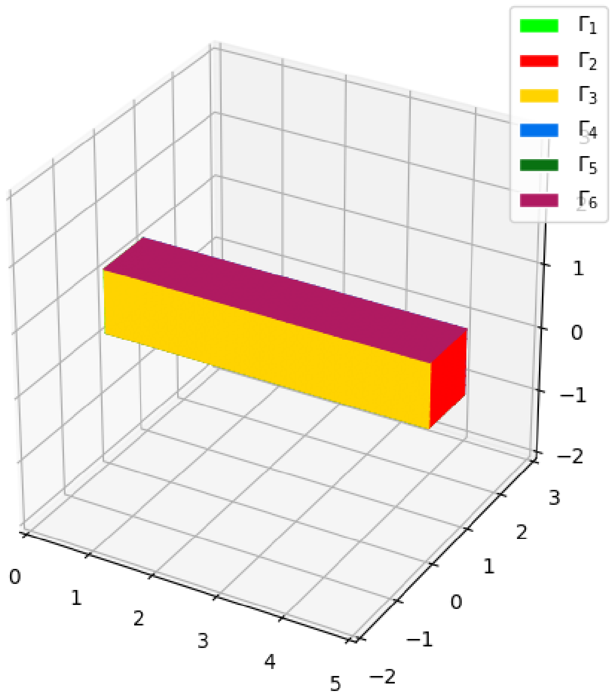


Figure 5.3: Domain and boundaries

We want to solve the elasticity problem (5.1) to (5.3) with $\Gamma^D = \Gamma_1$, $\Gamma^N = \bigcup_{i=2}^{6}\Gamma_i$ and $\boldsymbol{f} = \boldsymbol{x} \mapsto (0,0,-1)^t$.

Listing 5.2: **3D elasticity, Python code**

```
mu= E/(2*(1+nu))
lam = E*nu/((1+nu)*(1-2*nu))
gam=lam+2*mu
L=5;N=7
print('--------------------------------------------------------')
```

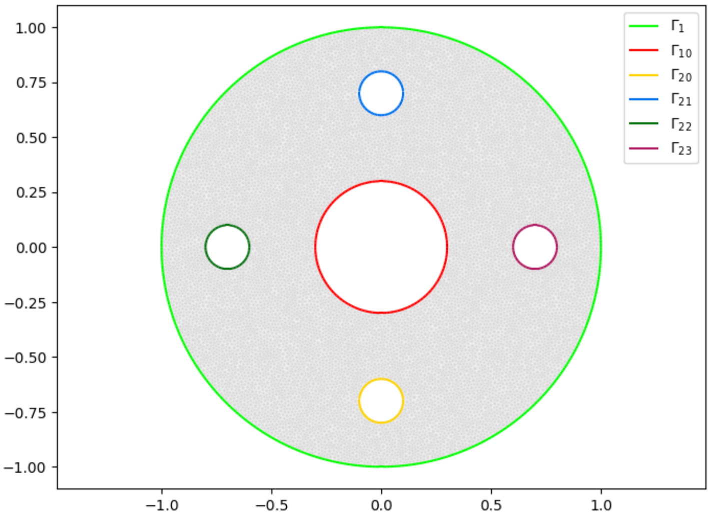


Figure 5.5: Domain and boundaries

```
Th=HyperCube(3,[L*N,N,N],mapping=mapping)
print('     Mesh sizes : nq=%d, nme=%d, h=%.3e'%(Th.nq,Th.get_nme(),Th.get_h()))
pde=PDE(Op=Hop, f=[0,0,-1])
bvp=BVP(Th,pde=pde)
bvp.setDirichlet(1,[0,0,0])
```

The displacement scaled by a factor 2000 for a given mesh is shown on Figure 5.4.

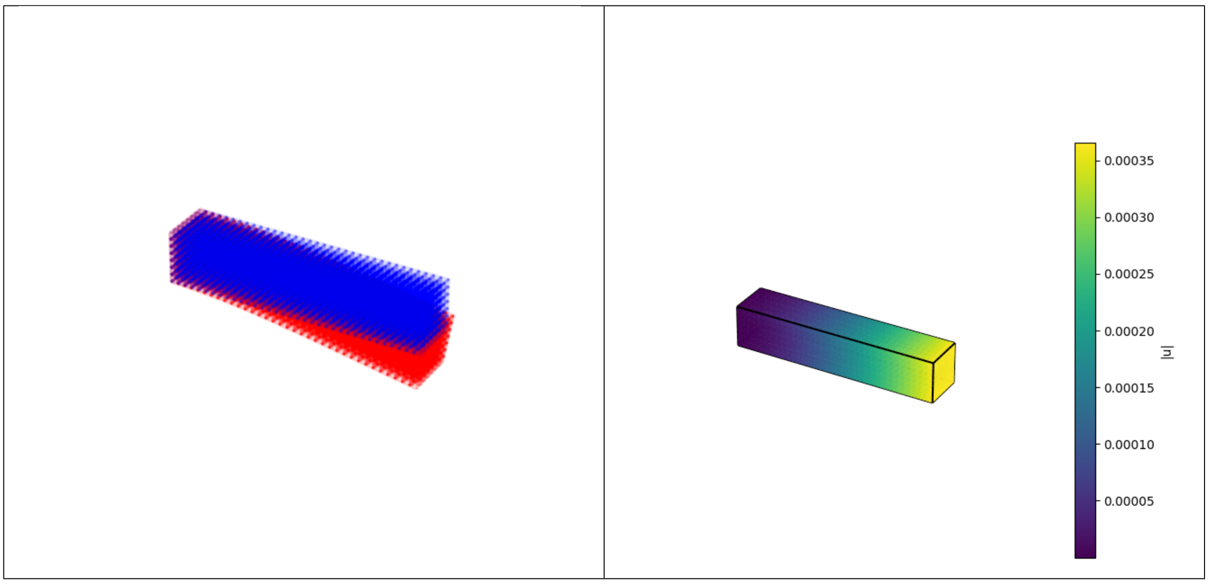


Figure 5.4: 3D elasticity problem: mesh displacement scaled by a factor 2000 (left) and norm of the displacement (right)

5.2 Stationary heat with potential flow in 2D

Let Γ_1 be the unit circle, Γ_{10} be the circle with center point $(0,0)$ and radius 0.3. Let Γ_{20}, Γ_{21}, Γ_{22} and Γ_{23} be the circles with radius 0.1 and respectively with center point $(0,-0.7)$, $(0,0.7)$, $(-0.7,0)$ and $(0.7,0)$. The domain $\Omega \subset \mathbb{R}^2$ is defined as the inner of Γ_1 and the outer of all other circles (see Figure 5.5).

The 2D problem to solve is the following

***Usual* BVP 6 : 2D problem : stationary heat with potential flow**

Find $u \in \mathrm{H}^2(\Omega)$ such that

$$\begin{aligned} -\operatorname{div}(\alpha \nabla u) + \langle \boldsymbol{V}, \nabla u \rangle + \beta u &= 0 \quad \text{in } \Omega \subset \mathbb{R}^2, && (5.10) \\ u &= 20 * \boldsymbol{x}_2 \quad \text{on } \Gamma_{21}, && (5.11) \\ u &= 0 \quad \text{on } \Gamma_{22} \cup \Gamma_{23}, && (5.12) \\ \frac{\partial u}{\partial n} &= 0 \quad \text{on } \Gamma_1 \cup \Gamma_{10} \cup \Gamma_{20} && (5.13) \end{aligned}$$

where Ω and its boundaries are given in Figure 5.5. This problem is well posed if $\alpha(\boldsymbol{x}) > 0$ and $\beta(\boldsymbol{x}) \geqslant 0$.
We choose α and β in Ω as :

$$\begin{aligned} \alpha(\boldsymbol{x}) &= 0.1 + \boldsymbol{x}_2^2, \\ \beta(\boldsymbol{x}) &= 0.01 \end{aligned}$$

The potential flow is the velocity field $\boldsymbol{V} = \nabla \phi$ where the scalar function ϕ is the velocity potential solution of the 2D BVP (5.14)-(5.17)

***Usual* BVP 7 : 2D velocity potential BVP**

Find $\phi \in \mathrm{H}^2(\Omega)$ such that

$$\begin{aligned} -\Delta \phi &= 0 \quad \text{in } \Omega, && (5.14) \\ \phi &= -20 \quad \text{on } \Gamma_{21}, && (5.15) \\ \phi &= 20 \quad \text{on } \Gamma_{20}, && (5.16) \\ \frac{\partial \phi}{\partial n} &= 0 \quad \text{on } \Gamma_1 \cup \Gamma_{23} \cup \Gamma_{22} && (5.17) \end{aligned}$$

Then the potential flow $\boldsymbol{V}$ is *solution* of (5.18)

***Usual* vector BVP 3 : 2D potential flow**

Find $\boldsymbol{V} = (\boldsymbol{V}_1, \boldsymbol{V}_2) \in \mathrm{H}^1(\Omega) \times \mathrm{H}^1(\Omega)$ such that

$$\boldsymbol{V} = \nabla \phi \quad \text{in } \Omega, \qquad (5.18)$$

For a given mesh, the numerical results are represented in Figure 5.6.

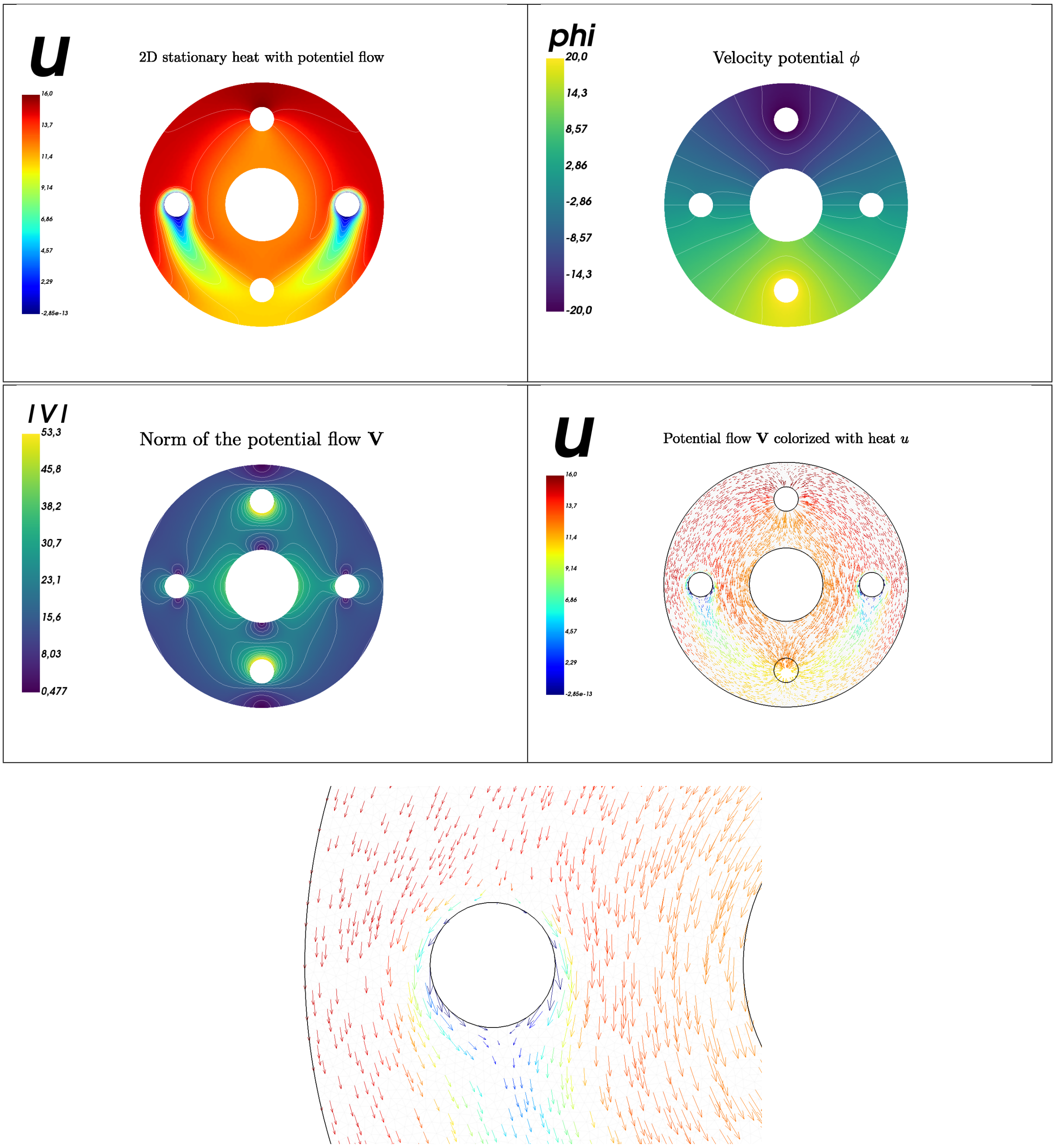


Figure 5.6: Heat u (top left), velocity potential ϕ (top right), norm of the potential flow (middle left), potential flow $\boldsymbol{V}$ colorized with heat u (middle right and bottom)

Now we will present two manners of solving these problems using FC-VFEM$\mathbb{P}_1$ codes.

5.2.1 Method 1 : split in three parts

The 2D potential velocity problem (5.14)-(5.17) can be equivalently expressed as the scalar BVP (1.2)-(1.4) :

Scalar BVP 9 : 2D potential velocity

Find $\phi \in \mathrm{H}^2(\Omega)$ such that

$$\begin{aligned} \mathcal{L}(\phi) &= f && \text{in } \Omega, \\ \phi &= g^D && \text{on } \Gamma^D, \\ \frac{\partial \phi}{\partial n_{\mathcal{L}}} + a^R \phi &= g^R && \text{on } \Gamma^R. \end{aligned}$$

where

- $\mathcal{L} := \mathcal{L}_{\mathbb{I},\mathbf{0},\mathbf{0},0}$, and then the conormal derivative of ϕ is given by

$$\frac{\partial \phi}{\partial n_{\mathcal{L}}} := \langle \mathbb{A} \nabla \phi, \boldsymbol{n} \rangle - \langle \boldsymbol{b} \phi, \boldsymbol{n} \rangle = \frac{\partial \phi}{\partial n}.$$

- $f(\boldsymbol{x}) := 0$
- $\Gamma^D = \Gamma_{20} \cup \Gamma_{21}$
- $\Gamma^R = \Gamma_1 \cup \Gamma_{23} \cup \Gamma_{22}$
- $g^D := 20$ on Γ_{20}, and $g^D := -20$ on Γ_{21}
- $g^R = a^R := 0$ on Γ^R. (Neumann boundary condition)

The code using the package for solving (5.14)-(5.17) is given in Listing 5.3.

Listing 5.3: **Stationary heat with potential flow in 2D, Python code (method 1)**

```
af=lambda x,y: 0.1+y**2
gD=lambda x,y: 20*y
b=0.01
geofile='disk5holes.geo'
Th=siMesh(meshfile)
print('     Mesh sizes : nq=%d, nme=%d, h=%.3e'%(Th.nq,Th.get_nme(),Th.get_h()))
bvpVelocityPotential=BVP(Th,pde=PDE(Op=Lop))
bvpVelocityPotential.setDirichlet(20,+20.)
bvpVelocityPotential.setDirichlet(21,-20.)
print('*** Solving 2D velocity potential BVP')
print('*** Setting 2D potential flow operator')
```

Now to compute $\boldsymbol{V}$, we can write the potential flow problem (5.18) with $\mathcal{H}$-operators as

$$\boldsymbol{V} = \begin{pmatrix} \boldsymbol{V}_1 \\ \boldsymbol{V}_2 \end{pmatrix} = \mathcal{B} \begin{pmatrix} \phi \\ \phi \end{pmatrix}$$

where

$$\mathcal{B} = \begin{pmatrix} \mathcal{L}_{\mathbb{O}_2,\mathbf{0}_2,(1,0)^t,1} & 0 \\ 0 & \mathcal{L}_{\mathbb{O}_2,\mathbf{0}_2,(0,1)^t,0} \end{pmatrix}$$

The code using the package for solving this problem is given in Listing 5.4.

Listing 5.4: **Stationary heat with potential flow in 2D, Python code (method 1)**

```
Hop.H[0][0]=Loperator(dim=dim,c=[1,0])
Hop.H[1][1]=Loperator(dim=dim,c=[0,1])
print('*** Applying 2D potential flow operator')
print('*** Setting 2D stationary heat BVP with potential flow')
```

Obviously, one can compute separately $\boldsymbol{V}_1$ and $\boldsymbol{V}_2$.

Finally, the stationary heat BVP (5.10)-(5.13) can be equivalently expressed as the scalar BVP (1.2)-(1.4) :

Usual BVP 8 : 2D stationary heat

Find $u \in \mathrm{H}^2(\Omega)$ such that

$$\begin{aligned} \mathcal{L}(u) &= f && \text{in } \Omega, \\ u &= g^D && \text{on } \Gamma^D, \\ \frac{\partial u}{\partial n_{\mathcal{L}}} + a^R u &= g^R && \text{on } \Gamma^R. \end{aligned}$$

where

- $\mathcal{L} := \mathcal{L}_{\begin{pmatrix}\alpha & 0\\ 0 & \alpha\end{pmatrix},\mathbf{0},\boldsymbol{V},\beta}$, and then the conormal derivative of u is given by

$$\frac{\partial u}{\partial n_{\mathcal{L}}} := \langle \mathbb{A}\boldsymbol{\nabla} u, \boldsymbol{n}\rangle - \langle \boldsymbol{b}u, \boldsymbol{n}\rangle = \alpha\frac{\partial u}{\partial n}.$$

- $f := 0$
- $\Gamma^D = \Gamma_{21} \cup \Gamma_{22} \cup \Gamma_{23}$
- $\Gamma^R = \Gamma_1 \cup \Gamma_{10} \cup \Gamma_{20}$
- $g^D(x, y) := 20y$ on Γ_{21}, and $g^D := 0$ on $\Gamma_{22} \cup \Gamma_{23}$
- $g^R := 0$ and $a^R := 0$ on Γ^R

The code using the package FC-VFEM$\mathbb{P}_1$ for solving (5.10)-(5.13) is given in Listing 5.6.

Listing 5.5: **Stationary heat with potential flow in 2D, Pythoncode (method 1)**

```
bvpHeat=BVP(Th,pde=PDE(Op=Lop))
bvpHeat.setDirichlet(21,gD)
bvpHeat.setDirichlet(22, 0)
bvpHeat.setDirichlet(23, 0)
print('*** Solving 2D stationary heat BVP with potential flow')
```

5.2.2 Method 2 : have fun with $\mathcal{H}$-operators

We can merged velocity potential BVP (5.14)-(5.17) and potential flow to obtain the new BVP

Usual vector BVP 4 : Velocity potential and potential flow in 2D

Find $\phi \in \mathrm{H}^2(\Omega)$ and $\boldsymbol{V} = (\boldsymbol{V}_1, \boldsymbol{V}_2) \in \mathrm{H}^1(\Omega) \times \mathrm{H}^1(\Omega)$ such that

$$-\left(\frac{\partial \boldsymbol{V}_1}{\partial x} + \frac{\partial \boldsymbol{V}_2}{\partial y}\right) = 0 \quad \text{in } \Omega, \tag{5.19}$$

$$\boldsymbol{V}_1 - \frac{\partial \phi}{\partial x} = 0 \quad \text{in } \Omega, \tag{5.20}$$

$$\boldsymbol{V}_2 - \frac{\partial \phi}{\partial y} = 0 \quad \text{in } \Omega, \tag{5.21}$$

$$\phi = -20 \quad \text{on } \Gamma_{21}, \tag{5.22}$$

$$\phi = 20 \quad \text{on } \Gamma_{20}, \tag{5.23}$$

$$\frac{\partial \phi}{\partial n} = 0 \quad \text{on } \Gamma_1 \cup \Gamma_{23} \cup \Gamma_{22} \tag{5.24}$$

We can also replace (5.19) by $-\Delta\phi = 0$.

Let $\boldsymbol{w} = \begin{pmatrix}\phi\\ \boldsymbol{V}_1\\ \boldsymbol{V}_2\end{pmatrix}$, the previous problem (5.19)-(5.24) can be equivalently expressed as the vector BVP (1.14)-(1.16) :

Vector BVP 4 : Velocity potential and potential flow in 2D

Find $\boldsymbol{w} = (\boldsymbol{w}_1, \boldsymbol{w}_2, \boldsymbol{w}_3) \in (\mathrm{H}^2(\Omega))^3$ such that

$$\mathcal{H}(\boldsymbol{w}) = \boldsymbol{f} \quad \text{in } \Omega, \tag{5.25}$$

$$\boldsymbol{w}_\alpha = g_\alpha^D \quad \text{on } \Gamma_\alpha^D,\ \forall \alpha \in [\![1,3]\!], \tag{5.26}$$

$$\frac{\partial \boldsymbol{w}}{\partial n_{\mathcal{H}_\alpha}} + a_\alpha^R \boldsymbol{w}_\alpha = g_\alpha^R \quad \text{on } \Gamma_\alpha^R,\ \forall \alpha \in [\![1,3]\!], \tag{5.27}$$

where $\Gamma_\alpha^R = \Gamma_\alpha^D = \varnothing$ for all $\alpha \in \{2,3\}$ (no boundary conditions on $\boldsymbol{V}_1$ and $\boldsymbol{V}_2$) and

- $\mathcal{H}$ is the 3-by-3 operator defined by

$$\mathcal{H} = \begin{pmatrix} 0 & \mathcal{L}_{\mathbb{O},-\boldsymbol{e}_1,\boldsymbol{0},0} & \mathcal{L}_{\mathbb{O},-\boldsymbol{e}_2,\boldsymbol{0},0} \\ \mathcal{L}_{\mathbb{O},\boldsymbol{0},-\boldsymbol{e}_1,0} & \mathcal{L}_{\mathbb{O},\boldsymbol{0},\boldsymbol{0},1} & 0 \\ \mathcal{L}_{\mathbb{O},\boldsymbol{0},-\boldsymbol{e}_2,0} & 0 & \mathcal{L}_{\mathbb{O},\boldsymbol{0},\boldsymbol{0},1} \end{pmatrix}$$

  its conormal derivative are given by

$$\frac{\partial \boldsymbol{w}_1}{\partial n_{\mathcal{H}_{1,1}}} = 0, \quad \frac{\partial \boldsymbol{w}_2}{\partial n_{\mathcal{H}_{1,2}}} = \boldsymbol{w}_2 \boldsymbol{n}_1, \quad \frac{\partial \boldsymbol{w}_3}{\partial n_{\mathcal{H}_{1,3}}} = \boldsymbol{w}_3 \boldsymbol{n}_2,$$

$$\frac{\partial \boldsymbol{w}_1}{\partial n_{\mathcal{H}_{2,1}}} = 0, \quad \frac{\partial \boldsymbol{w}_2}{\partial n_{\mathcal{H}_{2,2}}} = 0, \quad \frac{\partial \boldsymbol{w}_3}{\partial n_{\mathcal{H}_{2,3}}} = 0$$

$$\frac{\partial \boldsymbol{w}_1}{\partial n_{\mathcal{H}_{3,1}}} = 0, \quad \frac{\partial \boldsymbol{w}_2}{\partial n_{\mathcal{H}_{3,2}}} = 0, \quad \frac{\partial \boldsymbol{w}_3}{\partial n_{\mathcal{H}_{3,3}}} = 0.$$

  So we obtain

$$\frac{\partial \boldsymbol{w}}{\partial n_{\mathcal{H}_1}} \stackrel{\text{def}}{=} \sum_{\alpha=1}^{3} \frac{\partial \boldsymbol{w}_\alpha}{\partial n_{\mathcal{H}_{1,\alpha}}} = \langle \boldsymbol{V}, \boldsymbol{n} \rangle = \frac{\partial \phi}{\partial \boldsymbol{n}}, \tag{5.28}$$

  and

$$\frac{\partial \boldsymbol{w}}{\partial n_{\mathcal{H}_2}} = \frac{\partial \boldsymbol{w}}{\partial n_{\mathcal{H}_3}} := 0. \tag{5.29}$$

  From (5.29), we cannot impose boundary conditions on components 2 and 3.

- $\boldsymbol{f} := \boldsymbol{0}$
- $\Gamma_1^D = \Gamma_{20} \cup \Gamma_{21}$ and $\Gamma_1^R = \Gamma_1 \cup \Gamma_{10} \cup \Gamma_{22} \cup \Gamma_{23}$
- $g_1^D := 20$ on Γ_{20}, and $g_1^D := -20$ on Γ_{21}
- $g_1^R = a_1^R := 0$ on Γ_1^R

The solution of this vector BVP is obtain by using the Python code is given by Listing ??.

Listing 5.6: **Stationary heat with potential flow in 2D, Python code (method 1)**

```
Hop.H[0][1]=Loperator(dim=dim,b=[-1,0])
Hop.H[0][2]=Loperator(dim=dim,b=[0,-1])
Hop.H[1][0]=Loperator(dim=dim,c=[-1,0])
Hop.H[1][1]=Loperator(dim=dim,a0=1)
Hop.H[2][0]=Loperator(dim=dim,c=[0,-1])
Hop.H[2][2]=Loperator(dim=dim,a0=1)
bvpFlow=BVP(Th,pde=PDE(Op=Hop))
bvpFlow.setDirichlet(20,20,comps=[0])
bvpFlow.setDirichlet(21,-20,comps=[0])
print('*** Solving 2D potential velocity/flow BVP')
print('*** Setting 2D stationary heat BVP with potential flow')
phi=U[0]
Lop=Loperator(dim=d,d=d,A=[[af,None],[None,af]],c=V,a0=b)
```

5.3 Stationary heat with potential flow in 3D

Let $\Omega \subset \mathbb{R}^3$ be the cylinder given in Figure 5.7.

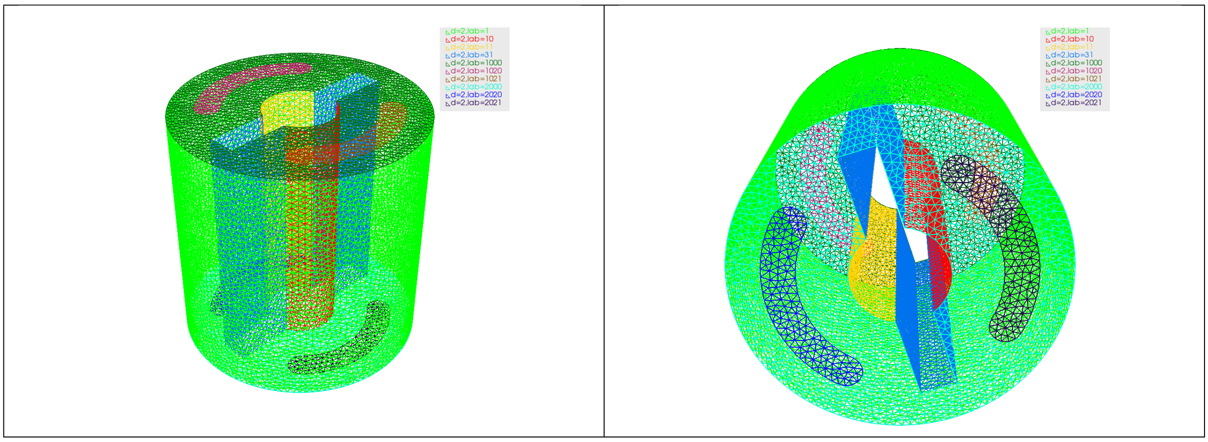


Figure 5.7: Stationary heat with potential flow : 3d mesh

The bottom and top faces of the cylinder are respectively $\Gamma_{1000} \cup \Gamma_{1020} \cup \Gamma_{1021}$ and $\Gamma_{2000} \cup \Gamma_{2020} \cup \Gamma_{2021}$. The hole surface is $\Gamma_{10} \cup \Gamma_{11} \cup \Gamma_{31}$ where $\Gamma_{10} \cup \Gamma_{11}$ is the cylinder part and Γ_{31} the plane part.

The 3D problem to solve is the following

***Usual* BVP 9 : 3D stationary heat with potential flow**

Find $u \in \mathrm{H}^2(\Omega)$ such that

$$\begin{aligned}
-\operatorname{div}(\alpha \boldsymbol{\nabla} u) + \langle \boldsymbol{V}, \boldsymbol{\nabla} u \rangle + \beta u &= 0 \quad \text{in } \Omega \subset \mathbb{R}^3, && (5.30)\\
u &= 30 \quad \text{on } \Gamma_{1020} \cup \Gamma_{2020}, && (5.31)\\
u &= 10\delta_{|z-1|>0.5} \quad \text{on } \Gamma_{10}, && (5.32)\\
\frac{\partial u}{\partial n} &= 0 \quad \text{otherwise} && (5.33)
\end{aligned}$$

where Ω and its boundaries are given in Figure 5.7. This problem is well posed if $\alpha(\boldsymbol{x}) > 0$ and $\beta(\boldsymbol{x}) \geqslant 0$.
We choose α and β in Ω as :

$$\begin{aligned}
\alpha(\boldsymbol{x}) &= 1 + (x_3 - 1)^2; ,\\
\beta(\boldsymbol{x}) &= 0.01
\end{aligned}$$

The potential flow is the velocity field $\boldsymbol{V} = \boldsymbol{\nabla} \phi$ where the scalar function ϕ is the velocity potential solution of the 3D BVP (5.34)-(5.37)

***Usual* BVP 10 : 3D velocity potential**

Find $\phi \in \mathrm{H}^1(\Omega)$ such that

$$\begin{aligned}
-\Delta \phi &= 0 \quad \text{in } \Omega, && (5.34)\\
\phi &= 1 \quad \text{on } \Gamma_{1021} \cup \Gamma_{2021}, && (5.35)\\
\phi &= -1 \quad \text{on } \Gamma_{1020} \cup \Gamma_{2020}, && (5.36)\\
\frac{\partial \phi}{\partial n} &= 0 \quad \text{otherwise} && (5.37)
\end{aligned}$$

Then the potential flow $\boldsymbol{V}$ is *solution* of (5.38)

Usual vector BVP 5 : 3D potential flow

Find $\boldsymbol{V} = (\boldsymbol{V}_1, \boldsymbol{V}_2, \boldsymbol{V}_3) \in \mathrm{H}^1(\Omega) \times \mathrm{H}^1(\Omega)$ such that

$$\boldsymbol{V} = \boldsymbol{\nabla}\phi \text{ in } \Omega, \tag{5.38}$$

For a given mesh, the numerical result for heat u is represented in Figure 5.8, velocity potential ϕ in Figure 5.9 and potential flow $\boldsymbol{V}$ are shown in Figure 5.10.

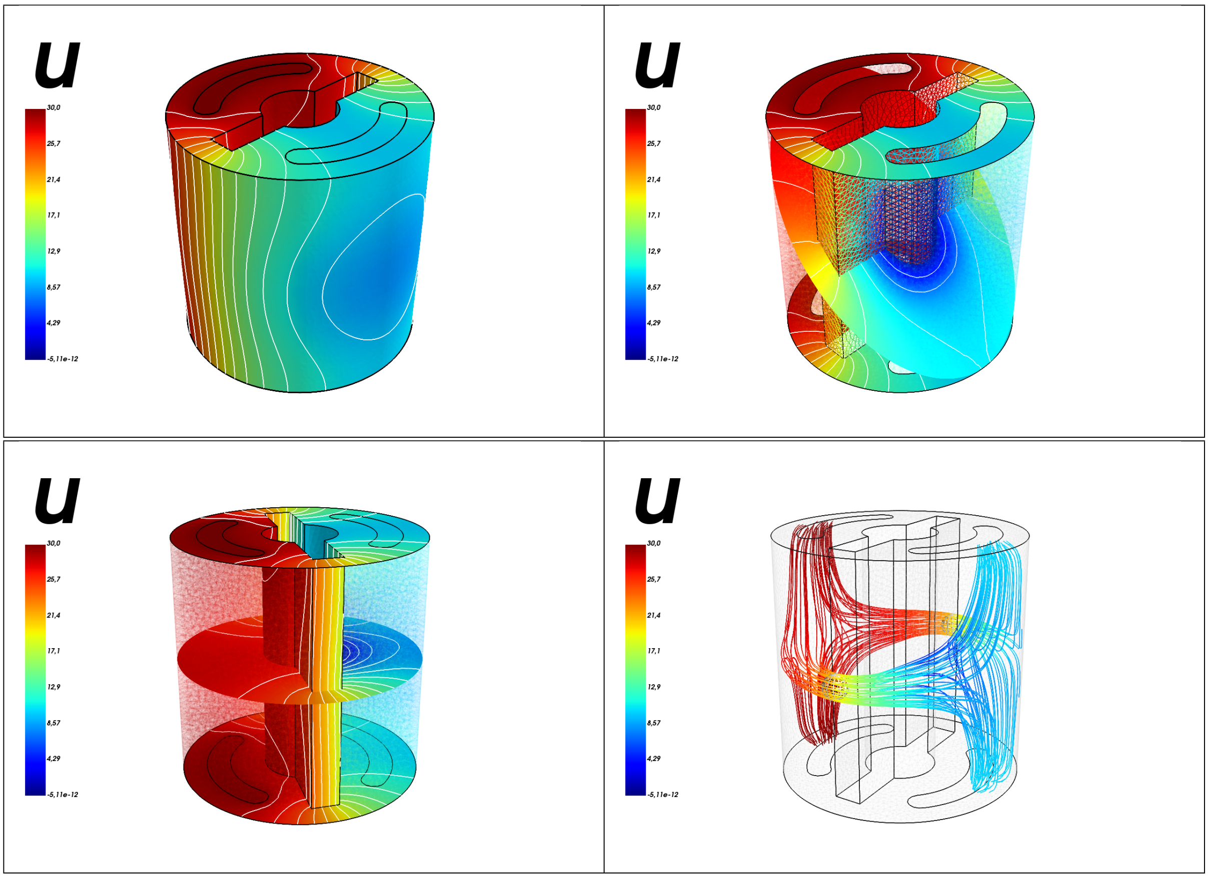


Figure 5.8: Heat solution u

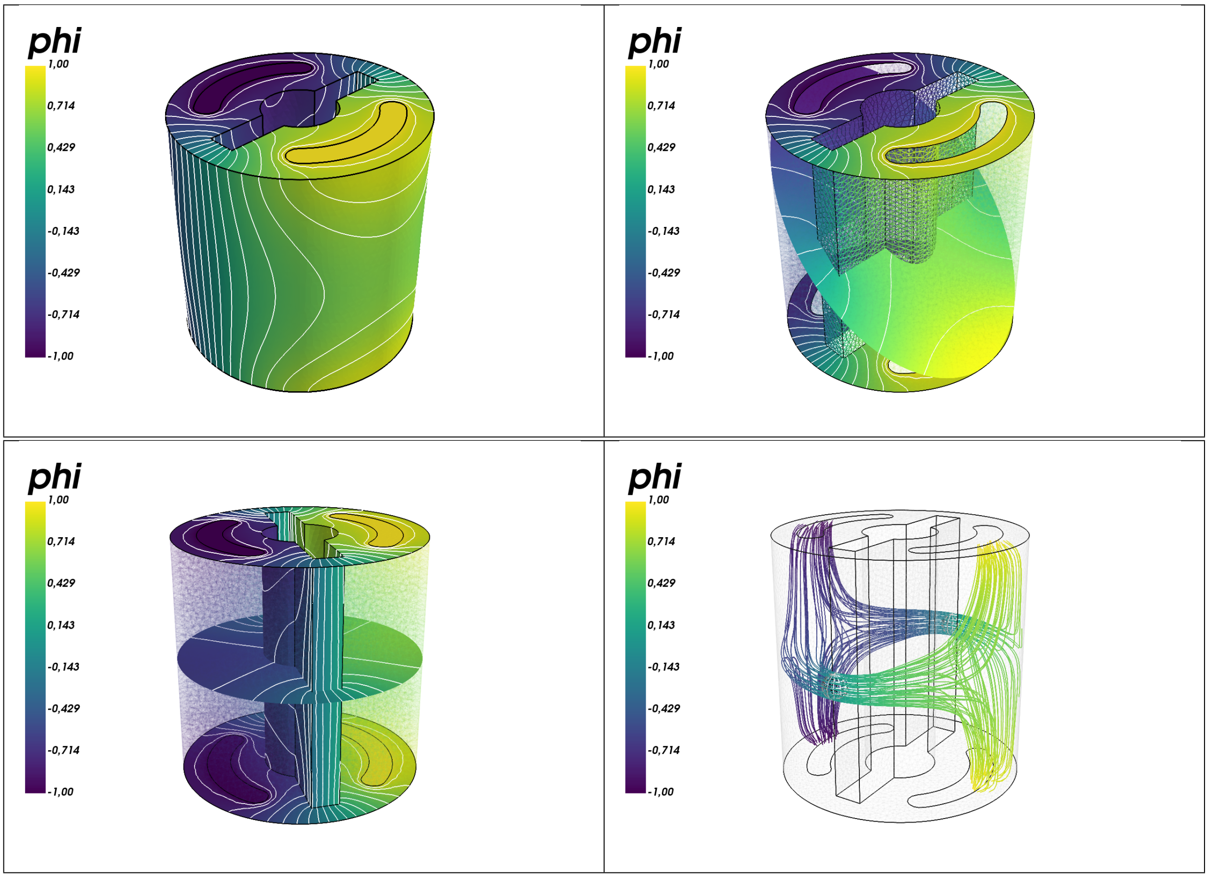


Figure 5.9: Velocity potential ϕ

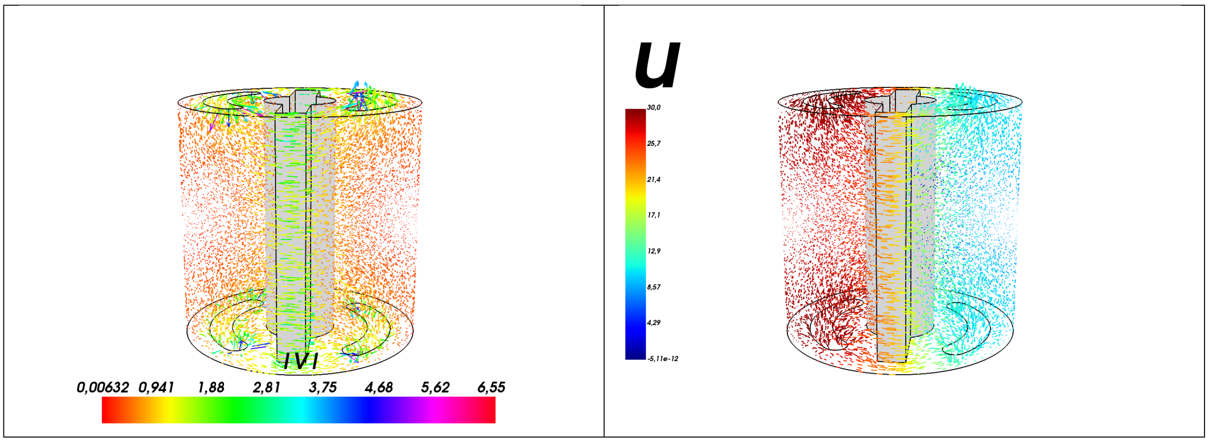


Figure 5.10: Potential flow $\boldsymbol{V}$

Now we will present two manners of solving these problems using FC-VFEM$\mathbb{P}_1$ codes.

5.3.1 Method 1 : split in three parts

The 3D potential velocity problem (5.34)-(5.37) can be equivalently expressed as the scalar BVP (1.2)-(1.4) :

Scalar BVP 10 : 3D potential velocity

Find $\phi \in \mathrm{H}^1(\Omega)$ such that

$$\begin{aligned} \mathcal{L}(\phi) &= f && \text{in } \Omega, \\ \phi &= g^D && \text{on } \Gamma^D, \\ \frac{\partial \phi}{\partial n_{\mathcal{L}}} + a^R \phi &= g^R && \text{on } \Gamma^R. \end{aligned}$$

where

- $\mathcal{L} := \mathcal{L}_{\mathbb{I},\mathbf{0},\mathbf{0},0}$, and then the conormal derivative of ϕ is given by

$$\frac{\partial \phi}{\partial n_{\mathcal{L}}} := \langle \mathbb{A} \nabla \phi, \boldsymbol{n} \rangle - \langle \boldsymbol{b}\phi, \boldsymbol{n} \rangle = \frac{\partial \phi}{\partial n}.$$

- $f(\boldsymbol{x}) := 0$
- $\Gamma^D = \Gamma_{1020} \cup \Gamma_{1021} \cup \Gamma_{2020} \cup \Gamma_{2021}$
- $\Gamma^R = \Gamma_1 \cup \Gamma_{10} \cup \Gamma_{11} \cup \Gamma_{31} \cup \Gamma_{1000} \cup \Gamma_{2000}$
- $g^D := 1$ on $\Gamma_{1021} \cup \Gamma_{2021}$, and $g^D := -1$ on $\Gamma_{1020} \cup \Gamma_{2020}$
- $g^R = a^R := 0$ on Γ^R. (Neumann boundary condition)

The code using the package for solving (5.34)-(5.37) is given in Listing **??**

Listing 5.7: **Stationary heat with potential flow in 3D, Python code (method 1)**

```
gD=lambda x,y,z: 10*(np.abs(z-1)>0.5)
a=lambda x,y,z: 1+(z-1)**2
b=0.01
geofile='cylinderkey.geo'
print('----------------------------------------------------------------')
bvpFlow=BVP(Th,pde=PDE(Op=Lop))
bvpFlow.setDirichlet(1021,1.)
bvpFlow.setDirichlet(2021,1.)
bvpFlow.setDirichlet(1020,-1.)
bvpFlow.setDirichlet(2020,-1.)
print('*** Solving 3D potential velocity/flow BVP')
```

Now to compute $\boldsymbol{V}$, we can write the potential flow problem (5.38)

- with $\mathcal{H}$-operators as

$$\boldsymbol{V} = \begin{pmatrix} \boldsymbol{V}_1 \\ \boldsymbol{V}_2 \\ \boldsymbol{V}_2 \end{pmatrix} = \mathcal{B} \begin{pmatrix} \phi \\ \phi \\ \phi \end{pmatrix}$$

where

$$\mathcal{B} = \begin{pmatrix} \mathcal{L}_{\mathbb{O}_3,\mathbf{0}_3,(1,0,0)^t,1} & 0 & 0 \\ 0 & \mathcal{L}_{\mathbb{O}_3,\mathbf{0}_3,(0,1,0)^t,0} & 0 \\ 0 & 0 & \mathcal{L}_{\mathbb{O}_3,\mathbf{0}_3,(0,0,1)^t,0} \end{pmatrix}$$

- with $\mathcal{L}$-operators as

$$\boldsymbol{V} = \begin{pmatrix} \boldsymbol{V}_1 \\ \boldsymbol{V}_2 \\ \boldsymbol{V}_2 \end{pmatrix} = \nabla \phi = \begin{pmatrix} \mathcal{L}_{\mathbb{O}_3,\mathbf{0}_3,(1,0,0)^t,0}(\phi) \\ \mathcal{L}_{\mathbb{O}_3,\mathbf{0}_3,(0,1,0)^t,0}(\phi) \\ \mathcal{L}_{\mathbb{O}_3,\mathbf{0}_3,(0,0,1)^t,0}(\phi) \end{pmatrix}$$

The code using FC-VFEM$\mathbb{P}_1$ package for solving this problem with $\mathcal{L}$-operators is given in Listing 5.8.

Listing 5.8: **Stationary heat with potential flow in 3D, Python code (method 1)**

```
Lop=Loperator(dim=dim,c=[1,0,0])
V[0]=Apply(Lop,Th,phi,solver=solver,perm=perm)
print('***  2) Computing V[1]')
V[1]=Apply(Lop,Th,phi,solver=solver,perm=perm)
print('***  3) Computing V[2]')
V[2]=Apply(Lop,Th,phi,solver=solver,perm=perm)
V=np.array(V)
print('*** Set 3D stationnary heat BVP with potential flow')
```

Finally, the stationary heat BVP (5.30)-(5.37) can be equivalently expressed as the scalar BVP (1.2)-(1.4) :

Scalar BVP 11 : 3D stationary heat

Find $u \in \mathrm{H}^1(\Omega)$ such that

$$\begin{aligned} \mathcal{L}(u) &= f && \text{in } \Omega, \\ u &= g^D && \text{on } \Gamma^D, \\ \frac{\partial u}{\partial n_{\mathcal{L}}} + a^R u &= g^R && \text{on } \Gamma^R. \end{aligned}$$

where

- $\mathcal{L} := \mathcal{L}_{\begin{pmatrix} \alpha & 0 & 0 \\ 0 & \alpha & 0 \\ 0 & 0 & \alpha \end{pmatrix}, \mathbf{0}, \boldsymbol{V}, \beta}$, and then the conormal derivative of u is given by

$$\frac{\partial u}{\partial n_{\mathcal{L}}} := \langle \mathbb{A} \boldsymbol{\nabla} u, \boldsymbol{n} \rangle - \langle \boldsymbol{b} u, \boldsymbol{n} \rangle = \alpha \frac{\partial u}{\partial n}.$$

- $f := 0$
- $\Gamma^D = \Gamma_{1020} \cup \Gamma_{2020} \cup \Gamma_{10}$
- $\Gamma^R = \Gamma_1 \cup \Gamma_{11} \cup \Gamma_{31} \cup \Gamma_{1000} \cup \Gamma_{1021} \cup \Gamma_{2000} \cup \Gamma_{2021}$
- $g^D(x, y, z) := 30$ on $\Gamma_{1020} \cup \Gamma_{2020}$, and $g^D(x, y, z) := 10(|z - 1| > 0.5)$ on Γ_{10}
- $g^R := 0$ and $a^R := 0$ on Γ^R

The code using the package for solving (5.30)-(**??**) is given in Figure 5.9.

Listing 5.9: **Stationary heat with potential flow in 3D, Python code (method 1)**

```
bvpHeat=BVP(Th,pde=PDE(Op=Lop))
bvpHeat.setDirichlet(1020,30.)
bvpHeat.setDirichlet(2020,30.)
bvpHeat.setDirichlet(10, gD)
print('*** Solve 3D stationnary heat BVP with potential flow')
```

5.3.2 Method 2 : have fun with $\mathcal{H}$-operators

To solve problem (5.30)-(5.33), we need to compute the velocity field $\boldsymbol{V}$. For that we can rewrite the potential flow problem (5.34)-(5.37), by introducing $\boldsymbol{V} = (\boldsymbol{V}_1, \boldsymbol{V}_2, \boldsymbol{V}_3)$ as unknowns :

Usual vector BVP 6 : Velocity potential and velocity field in 3D

Find $\phi \in H^2(\Omega)$ and $\boldsymbol{V} \in H^1(\Omega)^3$ such that

$$-\left(\frac{\partial \boldsymbol{V}_1}{\partial x} + \frac{\partial \boldsymbol{V}_2}{\partial y} + \frac{\partial \boldsymbol{V}_3}{\partial z}\right) = 0 \text{ in } \Omega, \tag{5.39}$$

$$\boldsymbol{V}_1 - \frac{\partial \phi}{\partial x} = 0 \text{ in } \Omega, \tag{5.40}$$

$$\boldsymbol{V}_2 - \frac{\partial \phi}{\partial y} = 0 \text{ in } \Omega, \tag{5.41}$$

$$\boldsymbol{V}_3 - \frac{\partial \phi}{\partial z} = 0 \text{ in } \Omega, \tag{5.42}$$

with boundary conditions (5.35) to (5.37).

We can also replace (5.39) by $-\Delta \phi = 0$.

Let $\boldsymbol{w} = \begin{pmatrix} \phi \\ \boldsymbol{V}_1 \\ \boldsymbol{V}_2 \\ \boldsymbol{V}_3 \end{pmatrix}$, the previous PDE can be written as a vector boundary value problem (see section 1.2) where the $\mathcal{H}$-operator is given by

$$\mathcal{H}(\boldsymbol{w}) = 0 \tag{5.43}$$

with

$$\mathcal{H}_{1,1} = 0, \quad \mathcal{H}_{1,2} = \mathcal{L}_{\mathbb{O},-\boldsymbol{e}_1,\boldsymbol{0},0}, \quad \mathcal{H}_{1,3} = \mathcal{L}_{\mathbb{O},-\boldsymbol{e}_2,\boldsymbol{0},0}, \quad \mathcal{H}_{1,4} = \mathcal{L}_{\mathbb{O},-\boldsymbol{e}_3,\boldsymbol{0},0}, \tag{5.44}$$

$$\mathcal{H}_{2,1} = \mathcal{L}_{\mathbb{O},\boldsymbol{0},-\boldsymbol{e}_1,0}, \quad \mathcal{H}_{2,2} = \mathcal{L}_{\mathbb{O},\boldsymbol{0},\boldsymbol{0},1}, \quad \mathcal{H}_{2,3} = 0, \quad \mathcal{H}_{2,4} = 0, \tag{5.45}$$

$$\mathcal{H}_{3,1} = \mathcal{L}_{\mathbb{O},\boldsymbol{0},-\boldsymbol{e}_2,0}, \quad \mathcal{H}_{3,2} = 0, \quad \mathcal{H}_{3,3} = \mathcal{L}_{\mathbb{O},\boldsymbol{0},\boldsymbol{0},1}, \quad \mathcal{H}_{3,4} = 0, \tag{5.46}$$

$$\mathcal{H}_{4,1} = \mathcal{L}_{\mathbb{O},\boldsymbol{0},-\boldsymbol{e}_3,0}, \quad \mathcal{H}_{4,2} = 0, \quad \mathcal{H}_{4,3} = 0, \quad \mathcal{H}_{4,4} = \mathcal{L}_{\mathbb{O},\boldsymbol{0},\boldsymbol{0},1}, \tag{5.47}$$

and $\boldsymbol{e}_1 = (1,0,0)^t$, $\boldsymbol{e}_2 = (0,1,0)^t$, $\boldsymbol{e}_3 = (0,0,1)^t$.

The conormal derivatives are given by

$$\frac{\partial \boldsymbol{w}_1}{\partial n_{\mathcal{H}_{1,1}}} = 0, \quad \frac{\partial \boldsymbol{w}_1}{\partial n_{\mathcal{H}_{2,1}}} = 0, \quad \frac{\partial \boldsymbol{w}_1}{\partial n_{\mathcal{H}_{3,1}}} = 0, \quad \frac{\partial \boldsymbol{w}_1}{\partial n_{\mathcal{H}_{4,1}}} = 0,$$

$$\frac{\partial \boldsymbol{w}_2}{\partial n_{\mathcal{H}_{1,2}}} = \boldsymbol{V}_1 \boldsymbol{n}_1, \quad \frac{\partial \boldsymbol{w}_2}{\partial n_{\mathcal{H}_{2,2}}} = 0, \quad \frac{\partial \boldsymbol{w}_2}{\partial n_{\mathcal{H}_{3,2}}} = 0, \quad \frac{\partial \boldsymbol{w}_2}{\partial n_{\mathcal{H}_{4,2}}} = 0,$$

$$\frac{\partial \boldsymbol{w}_3}{\partial n_{\mathcal{H}_{1,3}}} = \boldsymbol{V}_2 \boldsymbol{n}_2, \quad \frac{\partial \boldsymbol{w}_3}{\partial n_{\mathcal{H}_{2,3}}} = 0, \quad \frac{\partial \boldsymbol{w}_3}{\partial n_{\mathcal{H}_{3,3}}} = 0, \quad \frac{\partial \boldsymbol{w}_3}{\partial n_{\mathcal{H}_{4,3}}} = 0,$$

$$\frac{\partial \boldsymbol{w}_4}{\partial n_{\mathcal{H}_{1,4}}} = \boldsymbol{V}_3 \boldsymbol{n}_3, \quad \frac{\partial \boldsymbol{w}_4}{\partial n_{\mathcal{H}_{2,4}}} = 0, \quad \frac{\partial \boldsymbol{w}_4}{\partial n_{\mathcal{H}_{3,4}}} = 0, \quad \frac{\partial \boldsymbol{w}_4}{\partial n_{\mathcal{H}_{4,4}}} = 0,$$

So we obtain

$$\sum_{\alpha=1}^{4} \frac{\partial \boldsymbol{w}_\alpha}{\partial n_{\mathcal{H}_{1,\alpha}}} = \langle \boldsymbol{V}, \boldsymbol{n} \rangle = \langle \boldsymbol{\nabla} \phi, \boldsymbol{n} \rangle, \tag{5.48}$$

and

$$\sum_{\alpha=1}^{4} \frac{\partial \boldsymbol{w}_\alpha}{\partial n_{\mathcal{H}_{2,\alpha}}} = \sum_{\alpha=1}^{4} \frac{\partial \boldsymbol{w}_\alpha}{\partial n_{\mathcal{H}_{3,\alpha}}} = \sum_{\alpha=1}^{4} \frac{\partial \boldsymbol{w}_\alpha}{\partial n_{\mathcal{H}_{4,\alpha}}} = 0. \tag{5.49}$$

From (5.49), we cannot impose boundary conditions on components 2 to 4. Thus, with notation of section 1.2, we have $\Gamma_2^N = \Gamma_3^N = \Gamma_4^N = \Gamma$ with $g_2^N = g_3^N = g_4^N = 0$.

To take into account boundary conditions (5.35) to (5.37), we set $\Gamma_1^D = \Gamma_{1020} \cup \Gamma_{1021} \cup \Gamma_{2020} \cup \Gamma_{2021}$, $\Gamma_1^N = \Gamma \backslash \Gamma_1^D$ and $g_1^D = \delta_{\Gamma_{1020} \cup \Gamma_{2020}} - \delta_{\Gamma_{1021} \cup \Gamma_{2021}}$, $g_1^N = 0$.

The operator in (5.30) is given by $\mathcal{L}_{\alpha\mathbb{I},\boldsymbol{0},\boldsymbol{V},\beta}$. The conormal derivative $\frac{\partial u}{\partial n_{\mathcal{L}}}$ is

$$\frac{\partial u}{\partial n_{\mathcal{L}}} := \langle \mathbb{A} \boldsymbol{\nabla} u, \boldsymbol{n} \rangle - \langle \boldsymbol{b} u, \boldsymbol{n} \rangle = \alpha \frac{\partial u}{\partial n}.$$

The code using the package for solving (5.39)-(5.42) is given in Listing 5.10

Listing 5.10: **Stationary heat with potential flow in 3D, Python code (method 2)**

```
Hop.H[0][1]=Loperator(dim=dim,b=[-1,0,0])
Hop.H[0][2]=Loperator(dim=dim,b=[0,-1,0])
Hop.H[0][3]=Loperator(dim=dim,b=[0,0,-1])
Hop.H[1][0]=Loperator(dim=dim,c=[-1,0,0])
Hop.H[1][1]=Loperator(dim=dim,a0=1)
Hop.H[2][0]=Loperator(dim=dim,c=[0,-1,0])
Hop.H[2][2]=Loperator(dim=dim,a0=1)
Hop.H[3][0]=Loperator(dim=dim,c=[0,0,-1])
Hop.H[3][3]=Loperator(dim=dim,a0=1)
bvp=BVP(Th,pde=PDE(Op=Hop))
bvp.setDirichlet(1021,1.,comps=[0])
bvp.setDirichlet(2021,1.,comps=[0])
bvp.setDirichlet(1020,-1.,comps=[0])
bvp.setDirichlet(2020,-1.,comps=[0])
print('*** Solving 3D potential velocity/flow BVP')
V=np.array([U[1],U[2],U[3]])
phi=U[0]
print('*** Set 3D stationnary heat BVP with potential flow')
```

GIT : commit a284b9f9ae53f91022fefbd9a91a8d972746edee Date: Fri Jun 23 06:58:37 2017 +0200