A generic way to solve partial differential equations with P1-finite elements in vector languages : User guide

François Cuvelier Gilles Scarella

2015/06/10

Contents

1 Description of the generic problems

The notations of [5] 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

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

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 [5] and we note that we can omit either $\operatorname{div}(\boldsymbol{b}u)$ or $\langle \nabla u, \boldsymbol{c} \rangle$ 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**

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

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

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

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

$$\left\{ \begin{array}{rcccl} \mathcal{H} & : & \left(\mathrm{H}^2(\Omega)\right)^m & \longrightarrow & \left(L^2(\Omega)\right)^m \\ & & \boldsymbol{u} = (\boldsymbol{u}_1, \ldots, \boldsymbol{u}_m) & \longmapsto & \boldsymbol{f} = (\boldsymbol{f}_1, \ldots, \boldsymbol{f}_m) \stackrel{\text{def}}{=} \mathcal{H}(\boldsymbol{u}) \end{array} \right. \tag{1.6}$$

where

$$\boldsymbol{f}_\alpha = \sum_{\beta=1}^{m} \mathcal{H}_{\alpha,\beta}(\boldsymbol{u}_\beta), \quad \forall \alpha \in [\![1, m]\!], \tag{1.7}$$

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

$$\mathcal{H}_{\alpha,\beta} \stackrel{\text{def}}{=} \mathcal{L}_{\mathbb{A}^{\alpha,\beta},\boldsymbol{b}^{\alpha,\beta},\boldsymbol{c}^{\alpha,\beta},a_0^{\alpha,\beta}} \tag{1.8}$$

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

$$\mathcal{H}(\boldsymbol{u}) = \begin{pmatrix} \mathcal{L}_{\mathbb{A}^{1,1},\boldsymbol{b}^{1,1},\boldsymbol{c}^{1,1},a_0^{1,1}} & \dots & \mathcal{L}_{\mathbb{A}^{1,m},\boldsymbol{b}^{1,m},\boldsymbol{c}^{1,m},a_0^{1,m}} \\ \vdots & \ddots & \vdots \\ \mathcal{L}_{\mathbb{A}^{m,1},\boldsymbol{b}^{m,1},\boldsymbol{c}^{m,1},a_0^{m,1}} & \dots & \mathcal{L}_{\mathbb{A}^{m,m},\boldsymbol{b}^{m,m},\boldsymbol{c}^{m,m},a_0^{m,m}} \end{pmatrix} \begin{pmatrix} \boldsymbol{u}_1 \\ \vdots \\ \boldsymbol{u}_m \end{pmatrix}. \tag{1.9}$$

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 $\Gamma_\alpha^D \cap \Gamma_\alpha^R = \varnothing$. 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**

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

$$\mathcal{H}(\boldsymbol{u}) = \boldsymbol{f} \qquad \text{in } \Omega, \tag{1.10}$$

$$\boldsymbol{u}_\alpha = g_\alpha^D \qquad \text{on } \Gamma_\alpha^D,\ \forall \alpha \in [\![1,m]\!], \tag{1.11}$$

$$\frac{\partial \boldsymbol{u}}{\partial n_{\mathcal{H}_\alpha}} + a_\alpha^R \boldsymbol{u}_\alpha = g_\alpha^R \qquad \text{on } \Gamma_\alpha^R,\ \forall \alpha \in [\![1,m]\!], \tag{1.12}$$

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

$$\frac{\partial \boldsymbol{u}}{\partial n_{\mathcal{H}_\alpha}} \stackrel{\text{def}}{=} \sum_{\beta=1}^m \frac{\partial \boldsymbol{u}_\beta}{\partial n_{\mathcal{H}_{\alpha,\beta}}} = \sum_{\beta=1}^m \left(\langle \mathbb{A}^{\alpha,\beta} \nabla \boldsymbol{u}_\beta, \boldsymbol{n} \rangle - \langle \boldsymbol{b}^{\alpha,\beta} \boldsymbol{u}_\beta, \boldsymbol{n} \rangle \right). \tag{1.13}$$

The boundary conditions (1.12) are the **Robin** boundary conditions and (1.11) 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 $\boldsymbol{u}_2 = g_2^D$.In the following of the report we will solve by a $\mathbb{P}_1$-Lagrange finite element method *scalar* BVP (1.2) to (1.4) and *vector* BVP (1.10) to (1.12) without additional restrictive assumption.

2 Boundary Value Problems

2.1 Main data structures

We suppose that Ω is equipped with a mesh $\mathcal{T}_h$ (locally conforming) where its elements are d-simplices . We denote by Ω_h the union of the elements belonging

to the mesh, $\Omega_h = \bigcup_{K\in\mathcal{T}_h} K$, and by Γ_h its boundary, $\Gamma_h = \partial\Omega_h$.

The following data structure is associated to the mesh $\mathcal{T}_h$ and employs many notations already used in FreeFEM++ (see [3, 4]).

Mesh structure associated to $\mathcal{T}_h$

d	:	integer space dimension
n_q	:	integer number of vertices
n_{me}	:	integer number of elements (d-simplices)
n_{be}	:	integer number of boundary elements ((d−1)-simplices)
q	:	d-by-n_q array of reals array of vertices coordinates
me	:	(d+1)-by-n_{me} array of integers connectivity array for **mesh elements**
be	:	d-by-n_{be} array of integers connectivity array for **boundary elements**
bel	:	1-by-n_{be} array of integers array of boundary elements labels
vols	:	1-by-n_{me} array of reals array of mesh elements volumes

More precisely

- $q(\nu, j)$ is the ν-th coordinate of the j-th vertex, $\nu \in \{1, \ldots, d\}$, $j \in \{1, \ldots, n_q\}$. The j-th vertex will be also denoted by $q^j = q(:, j)$.
- $me(\beta, k)$ is the storage index of the β-th vertex of the k-th element (d-simplex), in the array q, for $\beta \in \{1, ..., d + 1\}$ and $k \in \{1, \ldots, n_{me}\}$. So $q(:, me(\beta, k))$ represents the coordinates of the β-th vertex of the k-th mesh element.
- $be(\beta, l)$ is the storage index of the β-th vertex of the l-th boundary element ((d−1)-simplex), in the array q, for $\beta \in \{1, ..., d\}$ and $l \in \{1, \ldots, n_{be}\}$. So $q(:, be(\beta, l))$ represents the coordinates of the β-th vertex of the l-th boundary element.
- $vols(k)$ is the volume of the k-th d-simplex .

A PDE structure contains all the data necessary to solve a scalar or vector BVP is described by

PDE structure

d	:	integer space dimension
m	:	integer operator dimension (1 for $\mathcal{L}$ operator)
op	:	operator $\mathcal{L}$ operator or $\mathcal{H}$ operator
f	:	1-by-m array of functions
$\mathcal{T}_h$	:	mesh structure
$\mathcal{B}_h$	:	1-by-nlab array of boundary mesh structures
labels	:	1-by-nlab array of integers Boundary mesh labels
nlab	:	integer number of boundary labels
bclR	:	m-by-nlab array of `BCrobin` structures
bclD	:	m-by-nlab array of `BCdirichlet` structures

2.2 First level functions or commonly used functions

We briefly describe the main functions that will be used in the sequel.

$\mathcal{T}_h \leftarrow$ **getMesh**(`FileName`) : to define the mesh $\mathcal{T}_h$ by reading a 2d or 3d mesh from the file `FileName`.

$\mathcal{T}_h \leftarrow$ **HyperCube**$(d, N, < trans = \Phi >)$: to define the mesh $\mathcal{T}_h$ as the unit hypercube $[0,1]^d$. There are $N(i)$ (or N if N is a scalar) points in each direction and the mesh of the hypercube contains $\prod_{i=1}^{d} N(i)$ points. Optionnal parameter `trans` set the displacement vector of mesh transformation $\Phi(\mathrm{q}) = [\Phi_1(\mathrm{q}), \ldots, \Phi_d(\mathrm{q})]$. The 2^d faces of the hypercube (before transformation) have an unique label : $\forall i \in [\![1, d]\!]$, the faces $x_i \equiv 0$ and $x_i \equiv 1$ are respectively of label $2i-1$ and $2i$.

`Lop` $\leftarrow$ **Loperator**$(d, \mathbb{A}, \boldsymbol{b}, \boldsymbol{c}, a_0)$: to initialize the operator $\mathcal{L}$ in dimension d given by (1.1) : `Lop` $\leftarrow \mathcal{L}_{\mathbb{A},\boldsymbol{b},\boldsymbol{c},a_0}$.

`Hop` $\leftarrow$ **Hoperator**(d, m) : to initialize the operator $\mathcal{H}$ given by (1.6) verifying $\mathcal{H}_{\alpha,\beta} = 0$, $\forall(\alpha,\beta) \in [\![1,m]\!]^2$. Each operator $\mathcal{H}_{\alpha,\beta}$ corresponds to `Hop.H`(α,β) and can be initialized by the function Loperator

pde $\leftarrow$ **initPDE**(`Op`, $\mathcal{T}_h$) : to initialize a PDE structure from an operator (either $\mathcal{L}$-operator or $\mathcal{H}$-operator) and a mesh. Default boundary conditions are homogeneous generalized Neumann.

pde $\leftarrow$ **setBC_PDE**(pde, `label`, `comps`, `type`, `g`, `ar`) : to define or modify the boundary conditions on the boundary $\Gamma_{\texttt{label}}$ on the mesh pde.$\mathcal{T}_h$ for components of index `comps` (in the scalar case `comps` $\equiv 1$). For a scalar PDE, we have for example

- Dirichlet condition : $\boldsymbol{u}_2 = g \;\; on \; \Gamma_{11}$, then
  pde $\leftarrow$ SETBC_PDE(pde, 11, 2, 'Dirichlet', $g, \varnothing$)
- generalized Neumann condition : $\frac{\partial \boldsymbol{u}}{\partial n_{\mathcal{H}_3}} = g \;\; on \; \Gamma_{12}$, then
  pde $\leftarrow$ SETBC_PDE(pde, 12, 3, 'Neumann', $g, \varnothing$)
- generalized Robin condition : $\frac{\partial \boldsymbol{u}}{\partial n_{\mathcal{H}_2}} + a_2^R \boldsymbol{u}_2 = g \;\; on \; \Gamma_{13}$, then
  pde $\leftarrow$ SETBC_PDE(pde, 13, 2, 'Robin', g, a_2^R)

$\boldsymbol{x} \leftarrow$ **SOLVEPDE**(pde) : to solve by $\mathbb{P}_1$-Lagrange finite elements the partial differential equation defined by the structure PDE. This function returns the solution $\boldsymbol{x}$ ()

2.3 Scalar case

2.3.1 Poisson PDE with mixed boundary conditions in 2D

We first consider the classical 2D Poisson problem with various boundary conditions. The problem to solve is the following

2D Poisson problem

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

$$
\begin{aligned}
-\Delta u &= f \;\text{ in } \Omega \subset \mathbb{R}^2, && (2.1)\\
u &= 0 \;\text{ on } \Gamma_1, && (2.2)\\
u &= 1 \;\text{ on } \Gamma_2, && (2.3)\\
\frac{\partial u}{\partial n} + a_R\, u &= -0.5 \;\text{ on } \Gamma_3, && (2.4)\\
\frac{\partial u}{\partial n} &= 0.5 \;\text{ on } \Gamma_4 && (2.5)
\end{aligned}
$$

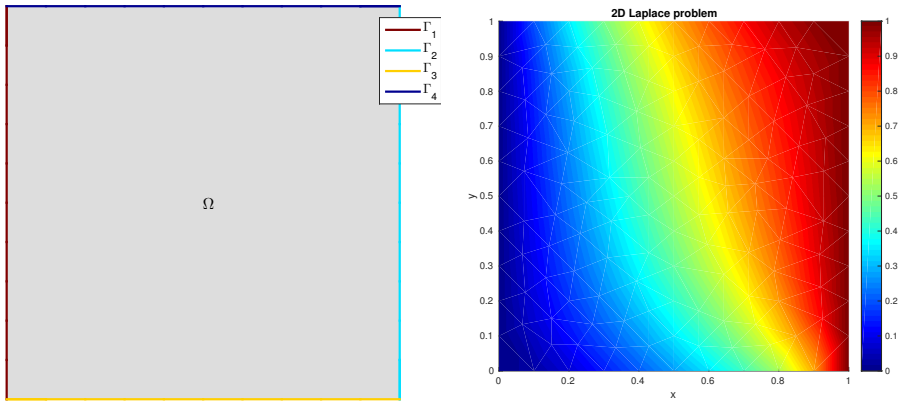
where $\Omega = [0,1]^2$ and its boundaries are given in Figure 1a.
f and a_R satisfy:

$$
\begin{aligned}
f(\boldsymbol{x}) &= \cos(\boldsymbol{x}_1 + \boldsymbol{x}_2) \quad \forall \boldsymbol{x} \in \Omega\\
a_R(\boldsymbol{x}) &= 1 + \boldsymbol{x}_1^2 + \boldsymbol{x}_2^2 \quad \forall \boldsymbol{x} \in \Omega
\end{aligned}
$$

The operator in (2.1) is the *Stiffness* operator : $\mathcal{L}_{\mathbb{I},\mathbf{0},\mathbf{0},0}$.
The conormal derivative $\frac{\partial u}{\partial n_{\mathcal{L}}}$ is

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

The algorithm using the toolbox for solving (2.1)-(2.5) is given in Algorithm 2.1. The corresponding Matlab/Octave and Python codes are given in Listing 1.



(a) Domain and boundaries (b) Result

Figure 1: Laplace2d01 problem

Algorithm 2.1 2D Poisson problem

1: $\mathcal{T}_h \leftarrow$ HyperCube$(2, 50)$ ▷ Build unit square mesh
2: Dop $\leftarrow$ Loperator$(2, \mathbb{I}, \mathbf{0}, \mathbf{0}, 0)$ ▷ Stiffness operator
3: PDE $\leftarrow$ initPDE$(\texttt{Dop}, \mathcal{T}_h)$
4: PDE $\leftarrow$ setBC_PDE$(\text{PDE}, 1, 1, \texttt{'Dirichlet'}, 0., \varnothing)$ ▷ $u = 0$ on Γ_1
5: PDE $\leftarrow$ setBC_PDE$(\text{PDE}, 2, 1, \texttt{'Dirichlet'}, 1., \varnothing)$ ▷ $u = 1$ on Γ_2
6: PDE $\leftarrow$ setBC_PDE$(\text{PDE}, 3, 1, \texttt{'Robin'}, -0.5, \boldsymbol{x} \to 1 + \boldsymbol{x}_1^2 + \boldsymbol{x}_2^2)$ ▷ $\frac{\partial u}{\partial n} + a_R\, u = -0.5$ on Γ_3
7: PDE $\leftarrow$ setBC_PDE$(\text{PDE}, 4, 1, \texttt{'Neumann'}, 0.5, \varnothing)$ ▷ $\frac{\partial u}{\partial n} = 0.5$ on Γ_4
8: PDE.$f \leftarrow (\boldsymbol{x}_1, \boldsymbol{x}_2) \mapsto \cos(\boldsymbol{x}_1 + \boldsymbol{x}_2)$
9: $\boldsymbol{u}_h \leftarrow$ SolvePDE(PDE)

A numerical solution for a given mesh is shown on Figure 1b

```
fprintf('1. Creating the mesh\n');
d=2;Th=HyperCube(d,50);
fprintf('2. Definition of the BVP\n');
LOp=Loperator(d,{1,0;0,1},[],[],[]);
PDE=initPDE(LOp,Th);
PDE=setBC_PDE(PDE,1,1,'Dirichlet',0);
PDE=setBC_PDE(PDE,2,1,'Dirichlet',1);
PDE=setBC_PDE(PDE,3,1,'Robin',-0.5 , ...
                @(x1,x2) 1+x1.^2+x2.^2 );
PDE=setBC_PDE(PDE,4,1,'Neumann',0.5);
PDE.f=@(x,y) cos(x+y) ;
fprintf('3. Solving BVP\n');
uh=solvePDE(PDE);
```

(a) Matlab/Octave

```
print('1. Creating the mesh')
d=2;Th=HyperCube(d,50)
print('2. Definition of the BVP')
LOp=Loperator(d=d,A=[[1,0],[0,1]])
pde=initPDE(LOp,Th)
pde=setBC_PDE(pde,1,0,'Dirichlet',0.,None)
pde=setBC_PDE(pde,2,0,'Dirichlet',1.,None)
pde=setBC_PDE(pde,3,0,'Robin',-0.5,
                lambda x,y: 1+x**2+y**2)
pde=setBC_PDE(pde,4,0,'Neumann',0.5,None)
pde.f=lambda x,y: cos(x+y)
print('3. Solving BVP');
uh=solvePDE(pde)
```

(b) Python

Listing 1: 2D Poisson codes

2.3.2 Poisson PDE with mixed boundary conditions in a 2D distorted domain

We first consider the classical Poisson problem with various boundary conditions in a 2D distorted domain. The problem to solve is the following

2D Poisson problem

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

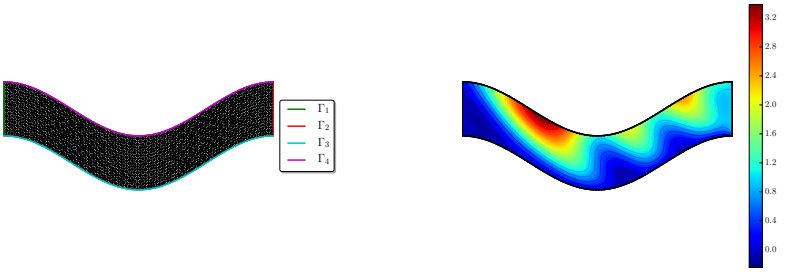
$$
\begin{aligned}
-\Delta u &= f \quad \text{in } \Omega \subset \mathbb{R}^2, && (2.6)\\
u &= 0 \quad \text{on } \Gamma_1, && (2.7)\\
u &= 1 \quad \text{on } \Gamma_2, && (2.8)\\
\frac{\partial u}{\partial n} + a_R\, u &= -0.5 \quad \text{on } \Gamma_3, && (2.9)\\
\frac{\partial u}{\partial n} &= 0.5 \quad \text{on } \Gamma_4 && (2.10)
\end{aligned}
$$

where Ω is the unit hypercube transformed by the function

$$\mathbf{\Phi}(x, y) = (20x,\ 2(2y - 1 + \cos(2\pi x)).$$

The boundaries are given in Figure 2a.
f and a_R satisfy:

$$
\begin{aligned}
f(\boldsymbol{x}) &= \cos(\boldsymbol{x}_1 + \boldsymbol{x}_2) \quad \forall \boldsymbol{x} \in \Omega\\
a_R(\boldsymbol{x}) &= 1 + \boldsymbol{x}_1^2 + \boldsymbol{x}_2^2 \quad \forall \boldsymbol{x} \in \Omega
\end{aligned}
$$



(a) Domain and boundaries (b) Result

Figure 2: Laplace2d01 problem

The operator in (2.6) is the *Stiffness* operator : $\mathcal{L}_{\mathbb{I},\mathbf{0},\mathbf{0},0}$.
The conormal derivative $\frac{\partial u}{\partial n_{\mathcal{L}}}$ is

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

The algorithm using the toolbox for solving (2.6)-(2.10) is given in Algorithm 2.2. The corresponding Matlab/Octave and Python codes are given in Listing 2.

Algorithm 2.2 2D Poisson problem in a distorted domain

1: $\mathcal{T}_h \leftarrow$ HyperCube$(2, [100, 20], trans = (x, y) \mapsto (20x,\ 2(2y - 1 + \cos(2\pi x)))$
2: Dop $\leftarrow$ Loperator$(2, \mathbb{I}, \mathbf{0}, \mathbf{0}, 0)$ ▷ Stiffness operator
3: PDE $\leftarrow$ initPDE$(\texttt{Dop}, \mathcal{T}_h)$
4: PDE $\leftarrow$ setBC_PDE$(\text{PDE}, 1, 1, \texttt{'Dirichlet'}, 0., \varnothing)$ ▷ $u = 0$ on Γ_1
5: PDE $\leftarrow$ setBC_PDE$(\text{PDE}, 2, 1, \texttt{'Dirichlet'}, 1., \varnothing)$ ▷ $u = 1$ on Γ_2
6: PDE $\leftarrow$ setBC_PDE$(\text{PDE}, 3, 1, \texttt{'Robin'}, -0.5, \boldsymbol{x} \rightarrow 1 + \boldsymbol{x}_1^2 + \boldsymbol{x}_2^2)$ ▷ $\frac{\partial u}{\partial n} + a_R\, u = -0.5$ on Γ_3
7: PDE $\leftarrow$ setBC_PDE$(\text{PDE}, 4, 1, \texttt{'Neumann'}, 0.5, \varnothing)$ ▷ $\frac{\partial u}{\partial n} = 0.5$ on Γ_4
8: PDE.$f \leftarrow (\boldsymbol{x}_1, \boldsymbol{x}_2) \mapsto \cos(\boldsymbol{x}_1 + \boldsymbol{x}_2)$
9: $\boldsymbol{u}_h \leftarrow$ SolvePDE(PDE)

A numerical solution for a given mesh is shown on Figure 2b

```
fprintf('1. Creating the mesh\n');
trans=@(q) [20*q(1,:); ...
           2*(2*q(2,:)-1+cos(2*pi*q(1,:)))];
Th=HyperCube(2,[100,20],trans);
fprintf('2. Definition of the BVP\n');
Lop=Loperator(2,{1,0;0,1},[],[],[]);
pde=initPDE(Lop,Th);
pde=setBC_PDE(pde,1,1,'Dirichlet', 0 );
pde=setBC_PDE(pde,2,1,'Dirichlet', 1 );
pde=setBC_PDE(pde,3,1,'Robin',-0.5 , ...
              @(x1,x2) 1+x1.^2+x2.^2 );
pde=setBC_PDE(pde,4,1,'Neumann', 0.5 );
pde.f=@(x,y) cos(x+y) ;
fprintf('3. Solving BVP\n');
uh=solvePDE(pde);
```

(a) Matlab/Octave

```
print('1. Creating the mesh')
trans=lambda q: np.c_[20*q[:,0],
      2*(2*q[:,1]-1+np.cos(2*pi*q[:,0]))]
Th=HyperCube(2,[100,20],trans=trans)
print('2. Definition of the BVP')
LOp=Loperator(d=2,A=[[1,0],[0,1]])
pde=initPDE(LOp,Th)
pde=setBC_PDE(pde,1,0,'Dirichlet',0.,None)
pde=setBC_PDE(pde,2,0,'Dirichlet',1.,None)
pde=setBC_PDE(pde,3,0,'Robin',-0.5,
              lambda x,y: 1+x**2+y**2)
pde=setBC_PDE(pde,4,0,'Neumann',0.5,None)
pde.f=lambda x,y: cos(x+y)
print('3. Solving BVP');
uh=solvePDE(pde)
```

(b) Python

Listing 2: 2D Poisson codes with distorted mesh

2.3.3 2D condenser problem

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

2D condenser problem

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

$$-\Delta u = 0 \text{ in } \Omega \subset \mathbb{R}^2, \tag{2.11}$$
$$u = 0 \text{ on } \Gamma_1, \tag{2.12}$$
$$u = -1 \text{ on } \Gamma_{98}, \tag{2.13}$$
$$u = 1 \text{ on } \Gamma_{99}, \tag{2.14}$$

where Ω and its boundaries are given in Figure 3a.

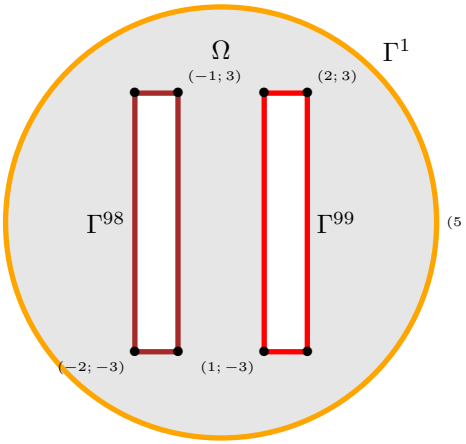
The problem (2.11)-(2.14) can be equivalently expressed as the scalar BVP (1.2)-(1.4) :

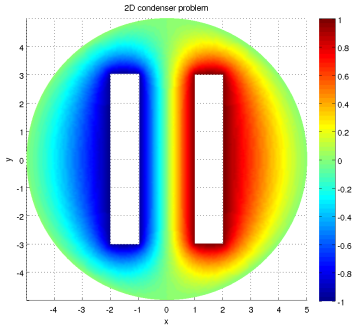
2D condenser problem as a *scalar* BVP

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

$$\mathcal{L}(u) = f \quad \text{in } \Omega,$$
$$u = g^D \quad \text{on } \Gamma^D.$$

where $\Gamma^R = \varnothing$ (no Robin boundary condition) and





(a) Domain for the condenser problem (b) Result for the 2D condenser problem

Figure 3: Condenser problem

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

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

- $f(\boldsymbol{x}) := 0$
- $\Gamma^D = \Gamma_1 \cup \Gamma_{98} \cup \Gamma_{99}$
- $g^D := 0$ on Γ_1, and $g^D := -1$ on Γ_{98} and $g^D := +1$ on Γ_{99}

The algorithm using the toolbox for solving (2.11)-(2.14) is the following:

Algorithm 2.3 2D condenser

1: $\mathcal{T}_h \leftarrow$ getMesh(...)	▷ Load FreeFEM++ mesh
2: Dop $\leftarrow$ Loperator($\mathbb{I}, \mathbf{0}, \mathbf{0}, 0$)	▷ Stiffness operator
3: PDE $\leftarrow$ initPDE(Dop, $\mathcal{T}_h$)	
4: PDE $\leftarrow$ setBCLabel(PDE, 'Dirichlet', 1, 1, 0.)	▷ $u = 0$ on Γ_1
5: PDE $\leftarrow$ setBCLabel(PDE, 'Dirichlet', 99, 1, 1.)	▷ $u = 1$ on Γ_{99}
6: PDE $\leftarrow$ setBCLabel(PDE, 'Dirichlet', 98, 1, −1.)	▷ $u = -1$ on Γ_{98}
7: $\boldsymbol{x} \leftarrow$ SolvePDE(PDE)	

We give respectively in Listing 1 and 2 the corresponding Matlab/Octave and Python codes.

Listing 1: **2D condenser, Matlab/Octave code**

```
fprintf('1. Reading of the condenser mesh\n')
Th=GetMesh2DOpt('condenser2D-10.msh');
fprintf('2. Definition of the BVP : 2D condenser\n')
Lop=Loperator(Th.d,{1,0;0,1},[],[],[]);
pde=initPDE(Lop,Th);
pde=setBC_PDE(pde, 1 ,1,'Dirichlet',0);
pde=setBC_PDE(pde,99 ,1,'Dirichlet',1);
pde=setBC_PDE(pde,98 ,1,'Dirichlet',-1);
fprintf('3. Solving BVP\n')
x=solvePDE(pde);
```

Listing 2: **2D condenser, Python code**

```
print("1. Reading of the condenser mesh")
Th=readFreeFEM('condenser2D-10.msh')
print("2. Definition of the BVP : 2D condenser")
Lop=Loperator(d=Th.d,A=[[1,None],[None,1]])
pde=initPDE(Lop,Th)
pde=setBC_PDE(pde,1,0,'Dirichlet',0.,None)
pde=setBC_PDE(pde,99,0,'Dirichlet',1.,None)
pde=setBC_PDE(pde,98,0,'Dirichlet',-1.,None)
print("3. Solving BVP")
x=solvePDE(pde)
```

The solution for a given mesh is shown on Figure 3b

2.3.4 Stationary convection-diffusion problem in 2D

The 2D problem to solve is the following

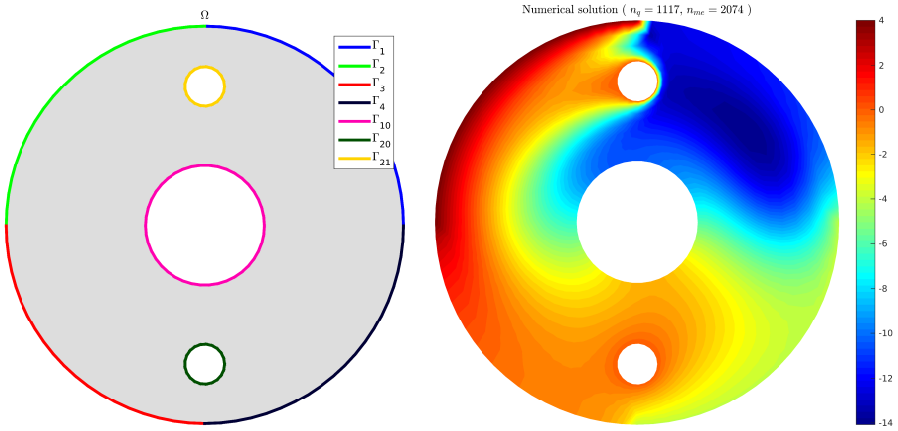
2D stationary convection-diffusion problem

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

$$
\begin{aligned}
-\operatorname{div}(\alpha \nabla u) + \langle \boldsymbol{V}, \nabla u \rangle + \beta u &= f \quad \text{in } \Omega \subset \mathbb{R}^2, && (2.15)\\
u &= 4 \quad \text{on } \Gamma_2, && (2.16)\\
u &= -4 \quad \text{on } \Gamma_4, && (2.17)\\
u &= 0 \quad \text{on } \Gamma_{20} \cup \Gamma_{21}, && (2.18)\\
\frac{\partial u}{\partial n} &= 0 \quad \text{on } \Gamma_1 \cup \Gamma_3 \cup \Gamma_{10} && (2.19)
\end{aligned}
$$

where Ω and its boundaries are given in Figure 4a. 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 :

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



(a) Domain and boundaries (b) Result

Figure 4: 2D stationary convection-diffusion problem

The problem (2.15)-(2.19) can be equivalently expressed as the scalar BVP (1.2)-(1.4) :

2D stationary convection-diffusion problem as a *scalar* BVP

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

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

where

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

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

- $\Gamma^D = \Gamma_2 \cup \Gamma_4 \cup \Gamma_{20} \cup \Gamma_{21}$ 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 (2.15)-(2.19) is the following:

Algorithm 2.4 Stationary convection-diffusion problem in 2D

$\mathcal{T}_h \leftarrow$ getMesh(...) ▷ Load FreeFEM++ mesh
$\alpha \leftarrow (x, y) \longmapsto 0.1 + (y - 0.5)(y - 0.5)$
$\beta \leftarrow 0.01$
$\mathcal{L} \leftarrow$ Loperator$(2, \begin{pmatrix} \alpha & 0 \\ 0 & \alpha \end{pmatrix}, \mathbf{0}, \begin{pmatrix} -10y \\ 10x \end{pmatrix}, \beta)$
pde $\leftarrow$ initPDE$(\mathcal{L}, \mathcal{T}_h)$ ▷ Set homogeneous `'Neumann'` condition on all boundaries
pde $\leftarrow$ setBC_PDE(pde, 2, 1, `'Dirichlet'`, 4., $\varnothing$) ▷ $u = 4$ on Γ_2
pde $\leftarrow$ setBC_PDE(pde, 4, 1, `'Dirichlet'`, -4, $\varnothing$) ▷ $u = -4$ on Γ_4
pde $\leftarrow$ setBC_PDE(pde, 20, 1, `'Dirichlet'`, 0, $\varnothing$ ▷ $u = 0$ on Γ_{20}
pde $\leftarrow$ setBC_PDE(pde, 21, 1, `'Dirichlet'`, 0, $\varnothing$) ▷ $u = 0$ on Γ_{21}
pde.$f \leftarrow (x, y) \mapsto -200 \exp(-10(x - 0.75)^2 + y^2)$
$\boldsymbol{x} \leftarrow$ SolvePDE(pde)

We give respectively in Listing 3 and 4 the corresponding Matlab/Octave and Python codes.

Listing 3: **2D stationary convection-diffusion, Matlab/Octave code**

```
fprintf('1. Reading of the mesh\n');
Th=GetMesh2DOpt('sampleD2d01-20.msh');
fprintf('2. Definition of the BVP\n')
af=@(x,y) 0.1+(y-0.5).*(y-0.5);
Vx=@(x,y) -10*y;Vy=@(x,y) 10*x;
b=0.01;g2=4;g4=-4;
f=@(x,y) -200.0*exp(-((x-0.75).^2+y.^2)/(0.1));
L=Loperator(Th.d,{af,[];[],af},[],{Vx,Vy},b);
pde=initPDE(L,Th);
pde=setBC_PDE(pde,2,1,'Dirichlet',g2);
pde=setBC_PDE(pde,4,1,'Dirichlet',g4);
pde=setBC_PDE(pde,20,1,'Dirichlet',0);
pde=setBC_PDE(pde,21,1,'Dirichlet',0);
pde.f=f;
fprintf('3. Solving BVP\n')
x=solvePDE(pde);
```

Listing 4: **2D stationary convection-diffusion, Python code**

```
print("1. Reading of the mesh")
Th=readFreeFEM("sampleD2d01-20.msh")
print("2. Definition of the BVP")
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*exp(-((x-0.75)**2+y**2)\
                       /(0.1));
Lop=Loperator(d=Th.d,A=[[af,None],[None,af]],
              c=[Vx,Vy],a0=b)
pde=initPDE(Lop,Th)
pde=setBC_PDE(pde,2,0, 'Dirichlet',g2)
pde=setBC_PDE(pde,4,0, 'Dirichlet',g4)
pde=setBC_PDE(pde,20,0,'Dirichlet',0)
pde=setBC_PDE(pde,21,0,'Dirichlet',0)
pde.f=f
print("3. Solving BVP")
x=solvePDE(pde)
```

The numerical solution for a given mesh is shown on Figure 4b

2.3.5 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}).

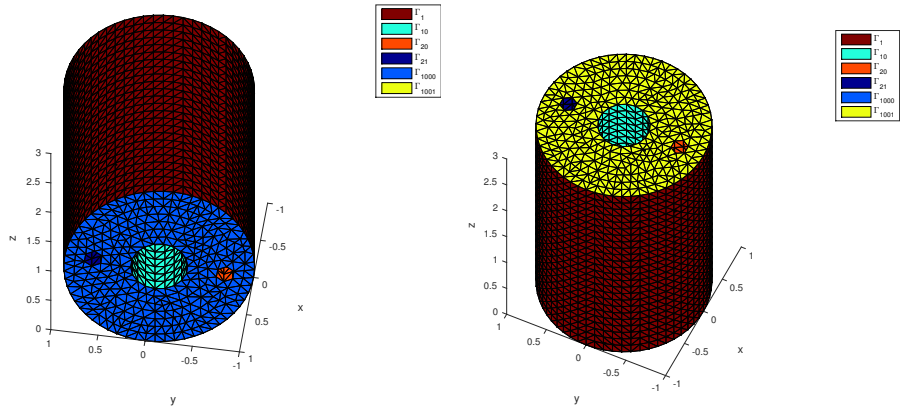
Let Ω be the cylinder defined by

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

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

$\Gamma_1, \Gamma_{10}, \Gamma_{20}$ 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 5.



(a) Bottom view (b) Top view

Figure 5: Mesh for the stationary convection-diffusion problem in 3D

The 3D problem to solve is the following

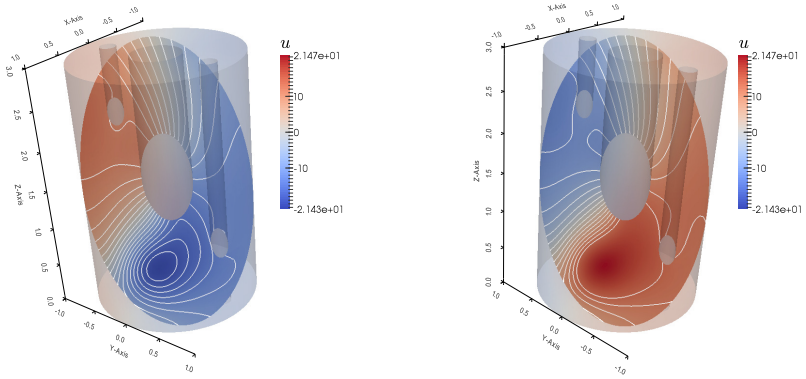
3D problem : Stationary convection-diffusion

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

$$
\begin{aligned}
-\operatorname{div}(\alpha \nabla u) + \langle \boldsymbol{V}, \nabla u \rangle + \beta u &= f \quad \text{in } \Omega \subset \mathbb{R}^3, && (2.20)\\
\alpha \frac{\partial u}{\partial n} + a_{20} u &= g_{20} \quad \text{on } \Gamma_{20}, && (2.21)\\
\alpha \frac{\partial u}{\partial n} + a_{21} u &= g_{21} \quad \text{on } \Gamma_{21}, && (2.22)\\
\frac{\partial u}{\partial n} &= 0 \quad \text{on } \Gamma^N && (2.23)
\end{aligned}
$$

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 :

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



(a) first view (b) second view

Figure 6: Stationary convection-diffusion problem in 3D : numerical solution

The problem (2.20)-(2.23) can be equivalently expressed as the scalar BVP (1.2)-(1.4) :

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

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

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

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

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

The algorithm using the toolbox for solving (2.20)-(2.23) is the following:

Algorithm 2.5 sampleD3d01 problem

$\mathcal{T}_h \leftarrow$ getMesh(...) ▷ Load FreeFEM++ 3D-mesh
$\alpha \leftarrow (x, y, z) \longmapsto 0.7 + z/10$
$\texttt{Dop} \leftarrow$ Loperator$(3, \begin{pmatrix} \alpha & 0 & 0 \\ 0 & \alpha & 0 \\ 0 & 0 & \alpha \end{pmatrix}, \mathbf{0}, \begin{pmatrix} -10y \\ 10x \\ 10z \end{pmatrix}, \beta)$
PDE $\leftarrow$ initPDE$(\texttt{Dop}, \mathcal{T}_h)$ ▷ Set homogeneous 'Neumann' condition on all boundaries
PDE $\leftarrow$ setBC_PDE(PDE, 20, 1, 'Robin', -0.05, 1.)
PDE $\leftarrow$ setBC_PDE(PDE, 21, 1, 'Robin', 0.05, 1.)
PDE.$f \leftarrow (x, y, z) \mapsto -800\exp(-10(x-0.65)^2 + y^2 + (z-0.5)^2)$
$\quad +800\exp(-10(x+0.65)^2 + y^2 + (z-0.5)^2)$
$\boldsymbol{x} \leftarrow$ SolvePDE(PDE)

We give respectively in Listing 5 and 6 the corresponding Matlab/Octave and Python codes.

Listing 5: **3D stationary convection-diffusion, Matlab/Octave code**

```
1
2  fprintf('1. Reading of the mesh\n');
3  Th=GetMesh3DOpt('sampleD3d01-6.mesh','format','medit');
4  fprintf('2. Definition of the BVP\n')
5  af=@(x,y,z) 0.7+z/10;%0.1+(y-0.5).*(y-0.5);
6  Vx=@(x,y,z) -10*y;Vy=@(x,y,z) 10*x;Vz=@(x,y,z) 10*z;
7  b=0.01;%0.01;g2=4;g4=-4;
8  f=@(x,y,z) -800.0*exp(-10*((x-0.65).*(x-0.65)+y.*y+(z-0.5).^2)) ...
9             +800.0*exp(-10*((x+0.65).*(x+0.65)+y.*y+(z-0.5).^2));
10 Lop=Loperator(Th.d,{af,[],[];[],af,[];[],[],af},[],{Vx,Vy,Vz},b);
11 pde=initPDE(Lop,Th);
12 pde=setBC_PDE(pde,20,1,'Robin', -0.05 , 1 );
13 pde=setBC_PDE(pde,21,1,'Robin', 0.05 , 1 );
14 pde.f=f;
15 fprintf('3. Solving BVP\n')
16 x=solvePDE(pde);
```

Listing 6: **3D stationary convection-diffusion, Python code**

```
print("1. Reading of the mesh")
Th=readFreeFEM3D("sampleD3d01-6.mesh")
print("2. Definition of the BVP")
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*exp(-10*((x-0.65)**2+y*y+(z-0.5)**2))+\
                 800.0*exp(-10*((x+0.65)**2+y*y+(z-0.5)**2))
Lop=Loperator(d=3,A=[[af,None,None],[None,af,None],[None,None,af]],\
              c=[Vx,Vy,Vz], a0=0.01)
pde=PDE(Lop,Th)
pde=setBC_PDE(pde,20,0,'Robin',-0.05,1)
pde=setBC_PDE(pde,21,0,'Robin',0.05,1)
pde.f=f
print("3. Solving BVP")
x=solvePDE(pde)
```

The numerical solution for a more refined mesh is shown on Figure 6

2.3.6 Laplace problem in $[0,1]^d$

The Laplace problem in any d-dimensional domain is considered here, with various boundary conditions.

Let $\Omega = [0,1]^d$ be the hypercube in $\mathbb{R}^d$. The 2^d faces of this hypercube have a unique label : $\forall i \in [\![1,d]\!]$, faces $x_i \equiv 0$ and $x_i \equiv 1$ are respectively of label $(2i-1)$ and $2i$.

Laplace problem in $[0,1]^d$

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

$$-\Delta u = 0 \quad \text{in } \Omega \subset \mathbb{R}^d, \tag{2.24}$$

$$u = 1 \quad \text{on } \Gamma_1 \cup \Gamma_2, \tag{2.25}$$

$$\frac{\partial u}{\partial n} + 5u = 1 \quad \text{on } \Gamma_3 \cup \Gamma_4, \tag{2.26}$$

$$\frac{\partial u}{\partial n} = 0 \quad \text{on } \bigcup_{i=3}^{d} \Gamma_{2i-1} \cup \Gamma_{2i}, \tag{2.27}$$

The problem (2.24)-(2.27) can be equivalently expressed as the scalar BVP (1.2)-(1.4) :

Laplace problem in $[0,1]^d$ as a *scalar* BVP

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

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

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

- $f := 0$

- $\Gamma^D = \Gamma_1 \cup \Gamma_2$ and $\Gamma^R = \bigcup_{i=2}^{d} \Gamma_{2i-1} \cup \Gamma_{2i}$
- $g^D := 1$ on $\Gamma_1 \cup \Gamma_2$
- $g^R := 1$ on $\Gamma_3 \cup \Gamma_4$ and $g^R := 0$ on $\bigcup_{i=3}^{d} \Gamma_{2i-1} \cup \Gamma_{2i}$
- $a^R(\boldsymbol{x}) := 1$ on $\Gamma_3 \cup \Gamma_4$ and $a^R := 0$ on $\bigcup_{i=3}^{d} \Gamma_{2i-1} \cup \Gamma_{2i}$

The algorithm using the toolbox for solving (2.24)-(2.27) is the following:

Algorithm 2.6 Laplace equation in $[0,1]^d$

1: $\mathcal{T}_h \leftarrow$ HyperCube(d, N)
2: $\mathcal{L} \leftarrow$ Loperator$(d, \mathbb{I}_d, \mathbf{0}_d, \mathbf{0}_d, 0)$ ▷ Stiffness operator
3: pde $\leftarrow$ initPDE$(\mathcal{L}, \mathcal{T}_h)$
4: pde $\leftarrow$ setBC_PDE(pde, 1, 1, 'Dirichlet', 1, $\varnothing$)
5: pde $\leftarrow$ setBC_PDE(pde, 2, 1, 'Dirichlet', 1, $\varnothing$)
6: pde $\leftarrow$ setBC_PDE(pde, 3, 1, 'Robin', 1, 1)
7: pde $\leftarrow$ setBC_PDE(pde, 4, 1, 'Robin', 1, 1)
8: $\boldsymbol{x} \leftarrow$ SolvePDE(pde)

The corresponding Matlab/Octave and Python codes are given in Listing 3.

```
fprintf('1. Set hypercube
    mesh\n');
d=input('d=');
N=input('N=');
Th=HyperCube(d,N);
fprintf('2. Definition of
    the BVP : %dD
    Laplace\n',d)
A=cell(d,d);
for i=1:d, A{i,i}=1;end
Lop=Loperator(d,A,[],[],[]);
pde=initPDE(Lop,Th);
pde=setBC_PDE(pde,1,1,'Dirichlet',1);
pde=setBC_PDE(pde,2,1,'Dirichlet',-1);
pde=setBC_PDE(pde,3,1,'Robin',5,1);
pde=setBC_PDE(pde,4,1,'Robin',5,1);
fprintf('3. Solving BVP\n')
x=solvePDE(pde);
```

(a) Matlab/Octave

```
print("1. Creating the mesh
    of the hypercube [0,1]^d")
d=int(input("d="))
N=int(input("N="))
Th=HyperCube(d,N)
print("2. Definition of the
    BVP : %dD Laplace"%d)
A=NoneMatrix(d,d)
for i in range(d):
  A[i][i]=1
Lop=Loperator(d=d,A=A)
pde=initPDE(Lop,Th)
pde=setBC_PDE(pde,1,0,'Dirichlet',0)
pde=setBC_PDE(pde,2,0,'Dirichlet',1)
pde=setBC_PDE(pde,3,0,'Robin',5,1)
pde=setBC_PDE(pde,4,0,'Robin',5,1)
print("3. Solving BVP")
x=solvePDE(pde)
```

(b) Python

Listing 3: Codes for Laplace problem in dimension d

It should be noted that in dimension d, the mesh of the hypercube $[0,1]^d$ obtained by the call to the function $\mathcal{T}_h \leftarrow$ HyperCube(d, N) contains $\mathrm{n_{me}} = d!(N-1)^d$ d-simplices , N being the number of points in each direction. This number can be very huge (see Table 1). So one does not need to be too ambitious in dimension $d > 3$ and choose a reasonable N.

$N \overset{\cdot}{\backslash} d$	2	3	4	5	6
5	32	384	6144	122880	2949120
10	162	4374	157464	7085880	382637520
15	392	16464	921984	64538880	2147483647

Table 1: Number of d-simplices in hypercube $[0,1]^d$ meshes

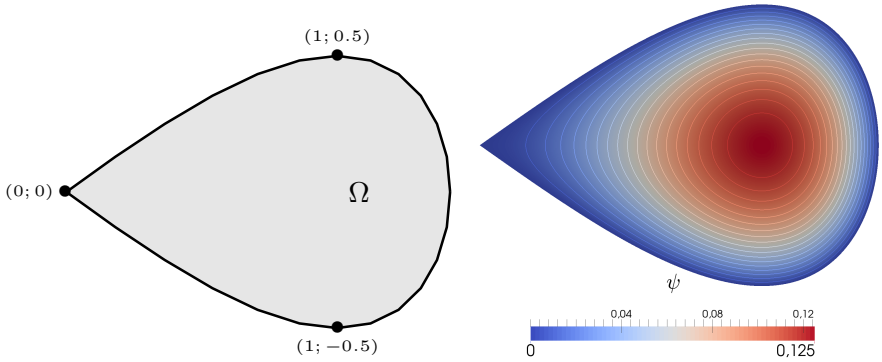
2.3.7 Grad-Shafranov problem

We consider here the 2D Grad-Shafranov problem already tested in [4] defined by

Grad-Shafranov problem

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

$$-\Delta\psi + a_0\,\psi = f \text{ in } \Omega \subset \mathbb{R}^2, \tag{2.28}$$
$$\psi = 0 \text{ on } \Gamma \tag{2.29}$$



(a) Domain (b) Result

Figure 7: Grad-Shafranov problem

The functions a_0 and f satisfy for all $\boldsymbol{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2) \in \Omega$:

$$a_0(\boldsymbol{x}) = \frac{1}{\boldsymbol{x}_1},$$
$$f(\boldsymbol{x}) = \boldsymbol{x}_1^2 + 1$$

The geometry can be described by a parametric function as

$$\begin{cases} \boldsymbol{x}_1(t) = \sqrt{1+\cos(t)} & \forall t \in [0, 2\pi], \\ \boldsymbol{x}_2(t) = 0.5\,\sin(t) \end{cases}$$

The operator in (2.28) is the following one : $\mathcal{L}_{\mathbb{I},\mathbf{0},\mathbf{0},a_0}$.

The conormal derivative $\frac{\partial\psi}{\partial n_{\mathcal{L}}}$ is

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

The algorithm using the toolbox for solving (2.28)-(2.29) is the following:

Algorithm 2.7 Grad-Shafranov problem

1: $\mathcal{T}_h \leftarrow$ getMesh(...) ▷ Load FreeFEM++ mesh
2: Dop $\leftarrow$ Loperator$(2, \mathbb{I}, \mathbf{0}, \mathbf{0}, \boldsymbol{x} \mapsto 1/\boldsymbol{x}_1)$
3: PDE $\leftarrow$ initPDE(Dop, $\mathcal{T}_h$)
4: PDE $\leftarrow$ setBC_PDE(PDE, 1, 1, 'Dirichlet', 0., $\varnothing$) ▷ $u = 0$ on Γ_1
5: PDE.$f \leftarrow \boldsymbol{x} \mapsto \boldsymbol{x}_1^2 + 1$
6: $\boldsymbol{x} \leftarrow$ SolvePDE(PDE)

The solution for a given mesh is shown on Figure 7.

2.4 Vector case

2.4.1 Elasticity problem

General case ($d = 2, 3$)

We consider here Hooke's law in linear elasticity, under small strain hypothesis (see for example [1]).

For a sufficiently regular vector field $\boldsymbol{u} = (u_1, \ldots, u_d) : \Omega \to \mathbb{R}^d$, 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:

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

The problem to solve is the following

Elasticity problem

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

$$-\operatorname{div}(\boldsymbol{\sigma}(\boldsymbol{u})) = \boldsymbol{f}, \quad \text{in } \Omega \subset \mathbb{R}^d, \tag{2.30}$$

$$\boldsymbol{\sigma}(\boldsymbol{u}).\boldsymbol{n} = \mathbf{0} \quad \text{on } \Gamma^R, \tag{2.31}$$

$$\boldsymbol{u} = \mathbf{0} \quad \text{on } \Gamma^D. \tag{2.32}$$

Now, with the following lemma, we obtain that this problem can be rewritten as the vector BVP defined by (1.10) to (1.12).

Lemma 1. *Let $\mathcal{H}$ be the d-by-d matrix of the second order linear differential operators defined in* (1.6) *where $\mathcal{H}_{\alpha,\beta} = \mathcal{L}_{\mathbb{A}^{\alpha,\beta},\mathbf{0},\mathbf{0},0}$, $\forall(\alpha,\beta) \in [\![1,d]\!]^2$, with*

$$(\mathbb{A}^{\alpha,\beta})_{k,l} = \mu\delta_{\alpha\beta}\delta_{kl} + \mu\delta_{k\beta}\delta_{l\alpha} + \lambda\delta_{k\alpha}\delta_{l\beta}, \ \forall(k,l) \in [\![1,d]\!]^2. \tag{2.33}$$

then

$$\mathcal{H}(\boldsymbol{u}) = -\operatorname{div}\boldsymbol{\sigma}(\boldsymbol{u}) \tag{2.34}$$

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

$$\frac{\partial \boldsymbol{u}}{\partial n_{\mathcal{H}_\alpha}} = (\boldsymbol{\sigma}(\boldsymbol{u}).\boldsymbol{n})_\alpha. \tag{2.35}$$

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

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 (1.6) 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 $d = 3$,
$$\mathbb{A}^{1,1} = \begin{pmatrix} \gamma & 0 & 0 \\ 0 & \mu & 0 \\ 0 & 0 & \mu \end{pmatrix},\quad \mathbb{A}^{1,2} = \begin{pmatrix} 0 & \lambda & 0 \\ \mu & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix},\quad \mathbb{A}^{1,3} = \begin{pmatrix} 0 & 0 & \lambda \\ 0 & 0 & 0 \\ \mu & 0 & 0 \end{pmatrix}$$
$$\mathbb{A}^{2,1} = \begin{pmatrix} 0 & \mu & 0 \\ \lambda & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix},\quad \mathbb{A}^{2,2} = \begin{pmatrix} \mu & 0 & 0 \\ 0 & \gamma & 0 \\ 0 & 0 & \mu \end{pmatrix},\quad \mathbb{A}^{2,3} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & \lambda \\ 0 & \mu & 0 \end{pmatrix},$$
$$\mathbb{A}^{3,1} = \begin{pmatrix} 0 & 0 & \mu \\ 0 & 0 & 0 \\ \lambda & 0 & 0 \end{pmatrix},\quad \mathbb{A}^{3,2} = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & \mu \\ 0 & \lambda & 0 \end{pmatrix},\quad \mathbb{A}^{3,3} = \begin{pmatrix} \mu & 0 & 0 \\ 0 & \mu & 0 \\ 0 & 0 & \gamma \end{pmatrix}.$$

The elasticity problem (2.30) to (2.32) can be rewritten as :
Find $\boldsymbol{u} = (\boldsymbol{u}_1, \ldots, \boldsymbol{u}_d) \in (\mathrm{H}^2(\Omega))^d$ such that

$$\mathcal{H}(\boldsymbol{u}) = \boldsymbol{f}, \qquad \text{in } \Omega, \tag{2.36}$$
$$\frac{\partial \boldsymbol{u}}{\partial n_{\mathcal{H}_\alpha}} = 0, \qquad \text{on } \Gamma^R_\alpha = \Gamma^R,\ \forall \alpha \in [\![1,d]\!] \tag{2.37}$$
$$\boldsymbol{u}_\alpha = 0, \qquad \text{on } \Gamma^D_\alpha = \Gamma^D,\ \forall \alpha \in [\![1,d]\!]. \tag{2.38}$$

2D example

For example, in 2d, we want to solve the elasticity problem (2.30) to (2.32) where Ω and its boundaries are given in Figure 8. We have $\Gamma^R = \Gamma^1 \cup \Gamma^2 \cup \Gamma^3$, $\Gamma^D = \Gamma^4$.

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 $\boldsymbol{f} = \boldsymbol{x} \mapsto (0,-1)^t$.

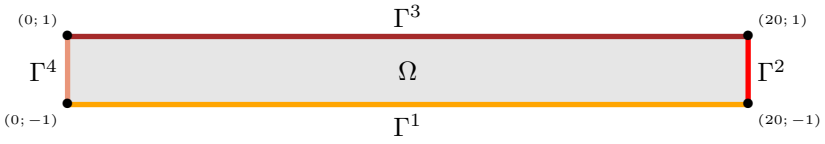


Figure 8: Domain for the 2D elasticity problem

The algorithm using the toolbox for solving (2.30)-(2.32) is the following:

Algorithm 2.8 2D elasticity

1: $\mathcal{T}_h \leftarrow$ getMesh(...) ▷ Load FreeFEM++ mesh
2: $\lambda \leftarrow \frac{E\nu}{(1+\nu)(1-2\nu)}$
3: $\mu \leftarrow \frac{E}{2(1+\nu)}$
4: Hop $\leftarrow$ InitHoperator$(2,2)$
5: Hop$(1,1) \leftarrow$ Loperator$(2, [2\mu+\lambda, 0; 0, \mu], \mathbf{0}, \mathbf{0}, 0)$
6: Hop$(2,1) \leftarrow$ Loperator$(2, [0, \lambda; \mu, 0], \mathbf{0}, \mathbf{0}, 0)$
7: Hop$(1,2) \leftarrow$ Loperator$(2, [0, \mu; \lambda, 0], \mathbf{0}, \mathbf{0}, 0)$
8: Hop$(2,2) \leftarrow$ Loperator$(2, [\mu, 0; 0, 2\mu+\lambda], \mathbf{0}, \mathbf{0}, 0)$
9: pde $\leftarrow$ initPDE(Hop, $\mathcal{T}_h$)
10: pde $\leftarrow$ setBC_PDE(pde, $4, 1:2,$ 'Dirichlet', $\boldsymbol{x} \rightarrow \mathbf{0}$)
11: pde.$f \leftarrow \boldsymbol{x} \rightarrow [0, -1]$
12: $\boldsymbol{x} \leftarrow$ SolvePDE(pde)

We give respectively in Listing 7 and 8 the corresponding Matlab/Octave and Python codes.

Listing 7: **2D elasticity, Matlab/Octave code**

```
fprintf('1. Reading of the mesh\n');
Th=GetMesh2DOpt('bar4-15.msh');
fprintf('2. Definition of the BVP\n');
E = 21.5e4; nu = 0.45;
mu= E/(2*(1+nu));
lambda = E*nu/((1+nu)*(1-2*nu));
gamma=lambda+2*mu;
Hop=Hoperator(2,2);
Hop.H{1,1}=Loperator(2,{gamma,[];[],mu},[],[],[]);
Hop.H{1,2}=Loperator(2,{[],lambda;mu,[]},[],[],[]);
Hop.H{2,1}=Loperator(2,{[],mu;lambda,[]},[],[],[]);
Hop.H{2,2}=Loperator(2,{mu,[];[],gamma},[],[],[]);
% One can also use the preset operator function
%   Hop=buildHoperator(2,2,'name','StiffElas','lambda',lambda,'mu',mu);
pde=initPDE(Hop,Th);
pde.f={0,-1};
pde=setBC_PDE(pde,4 ,1:2, 'Dirichlet',{0,0});
fprintf('3. Solving BVP\n');
x=solvePDE(pde);
```

Listing 8: **2D elasticity, Python code**

```
print('1. Reading of the mesh')
Th=readFreeFEM('bar4-15.msh')
print('2. Definition of the BVP')
E = 21.5e4; nu = 0.45
mu= E/(2*(1+nu))
lam = E*nu/((1+nu)*(1-2*nu))
gam=lam+2*mu
Hop=Hoperator(d=2,m=2)
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]])
# One can also use the preset operator function
#    Hop=StiffElasHoperators(2,lam,mu)
pde=initPDE(Hop,Th)
pde.f=[0,-1]
pde=setBC_PDE(pde,4,[0,1],'Dirichlet',[0,0],None)
print('3. Solving BVP')
x=solvePDE(pde,split=True)
```

For a given mesh, its displacement scaled by a factor 10 is shown on Figure 9

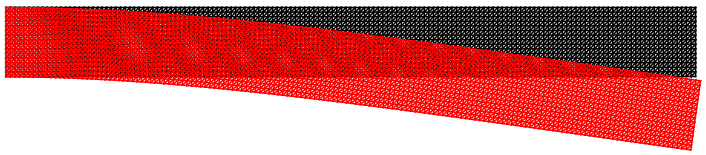


Figure 9: Mesh displacement scaled by a factor 10 for the 2D elasticity problem

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 10.

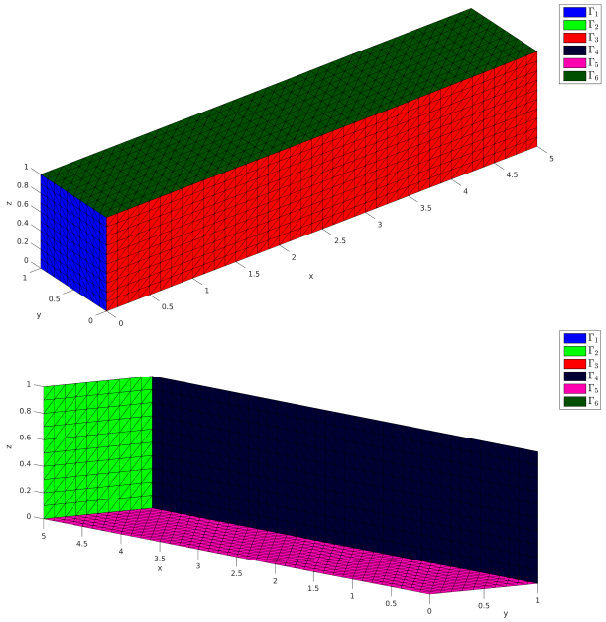


Figure 10: Domain Ω and boundaries for the 3D elasticity problem

We want to solve the elasticity problem (2.30) to (2.32) with $\Gamma^D = \Gamma_1$, $\Gamma^N = \bigcup_{i=2}^{6} \Gamma_i$ and $\boldsymbol{f} = \boldsymbol{x} \mapsto (0, 0, -1)^t$.

The algorithm using the toolbox for solving (2.30)-(2.32) is the following:

Algorithm 2.9 3D elasticity

$\mathcal{T}_h \leftarrow$ getMesh(...) ▷ Load FreeFEM++ mesh
$\lambda \leftarrow \frac{E\nu}{(1+\nu)(1-2\nu)}$
$\mu \leftarrow \frac{E}{2(1+\nu)}$
$\gamma \leftarrow \lambda + 2\mu$
Hop $\leftarrow$ Hoperator$(3, 3)$
Hop.H$(1, 1) \leftarrow$ Loperator$(3, [\gamma, 0, 0; 0, \mu, 0; 0, 0, \mu], \mathbf{0}, \mathbf{0}, 0)$
Hop.H$(1, 2) \leftarrow$ Loperator$(3, [0, \lambda, 0; \mu, 0, 0; 0, 0, 0], \mathbf{0}, \mathbf{0}, 0)$
Hop.H$(1, 3) \leftarrow$ Loperator$(3, [0, 0, \lambda; 0, 0, 0; \mu, 0, 0], \mathbf{0}, \mathbf{0}, 0)$
Hop.H$(2, 1) \leftarrow$ Loperator$(3, [0, \mu, 0; \lambda, 0, 0; 0, 0, 0], \mathbf{0}, \mathbf{0}, 0)$
Hop.H$(2, 2) \leftarrow$ Loperator$(3, [\mu, 0, 0; 0, \gamma, 0; 0, 0, \mu], \mathbf{0}, \mathbf{0}, 0)$
Hop.H$(2, 3) \leftarrow$ Loperator$(3, [0, 0, 0; 0, 0, \lambda; 0, \mu, 0], \mathbf{0}, \mathbf{0}, 0)$
Hop.H$(3, 1) \leftarrow$ Loperator$(3, [0, 0, \mu; 0, 0, 0; \lambda, 0, 0], \mathbf{0}, \mathbf{0}, 0)$
Hop.H$(3, 2) \leftarrow$ Loperator$(3, [0, 0, 0; 0, 0, \mu; 0, \lambda, 0], \mathbf{0}, \mathbf{0}, 0)$
Hop.H$(3, 3) \leftarrow$ Loperator$(3, [\mu, 0, 0; 0, \mu, 0; 0, 0, \gamma], \mathbf{0}, \mathbf{0}, 0)$
pde $\leftarrow$ initPDE$(\text{Hop}, \mathcal{T}_h)$
pde $\leftarrow$ setBC_PDE$(\text{pde}, 1, 1:3, \texttt{'Dirichlet'}, \boldsymbol{x} \to \mathbf{0})$
pde.$f \leftarrow \boldsymbol{x} \to [0, 0, -1]$
$\boldsymbol{x} \leftarrow$ SolvePDE(pde)

We give respectively in Listings 9 and 10 the corresponding Matlab/Octave and Python codes.

Listing 9: **3D elasticity, Matlab/Octave code**

```
fprintf('1. Reading of the mesh\n');
Th=GetMesh3DOpt('elasticity3D-10.mesh','format','medit');
fprintf('2. Definition of the BVP\n');
E = 21.5e4; nu = 0.29;
mu= E/(2*(1+nu));
lambda = E*nu/((1+nu)*(1-2*nu));
H=buildHoperator(3,3,'name','StiffElas','lambda',lambda,'mu',mu);
pde=initPDE(H,Th);
pde.f={0,0,-1};
pde=setBC_PDE(pde,1 ,1:3, 'Dirichlet',{0,0,0});
fprintf('3. Solving BVP\n');
x=solvePDE(pde);
```

Listing 10: **3D elasticity, Python code**

```
print('1. Reading of the mesh')
Th=readFreeFEM3D('elasticity3D-10.mesh')
print('2. Definition of the BVP')
E = 21.5e4; nu = 0.29
mu= E/(2*(1+nu))
lam = E*nu/((1+nu)*(1-2*nu))
gam=lam+2*mu
Hop=Hoperator(d=3,m=3)
Hop.H[0][0]=Loperator(d=3,A=[[gam,None,None],[None,mu,None],[None,None,mu]])
Hop.H[0][1]=Loperator(d=3,A=[[None,lam,None],[mu,None,None],[None,None,None]])
Hop.H[0][2]=Loperator(d=3,A=[[None,None,lam],[None,None,None],[mu,None,None]])
Hop.H[1][0]=Loperator(d=3,A=[[None,mu,None],[lam,None,None],[None,None,None]])
Hop.H[1][1]=Loperator(d=3,A=[[mu,None,None],[None,gam,None],[None,None,mu]])
Hop.H[1][2]=Loperator(d=3,A=[[None,None,None],[None,None,lam],[None,mu,None]])
Hop.H[2][0]=Loperator(d=3,A=[[None,None,mu],[None,None,None],[lam,None,None]])
Hop.H[2][1]=Loperator(d=3,A=[[None,None,None],[None,None,mu],[None,lam,None]])
Hop.H[2][2]=Loperator(d=3,A=[[mu,None,None],[None,mu,None],[None,None,gam]])
# One can also use the preset operator function
#    Hop=StiffElasHoperators(d,lam,mu)
pde=initPDE(Hop,Th)
pde.f=[0,0,-1]
pde=setBC_PDE(pde,1,[0,1,2],'Dirichlet',[0,0,0],None)
print('3. Solving BVP')
x=solvePDE(pde,split=True)
```

The displacement scaled by a factor 100 for a given mesh is shown on Figure 11.

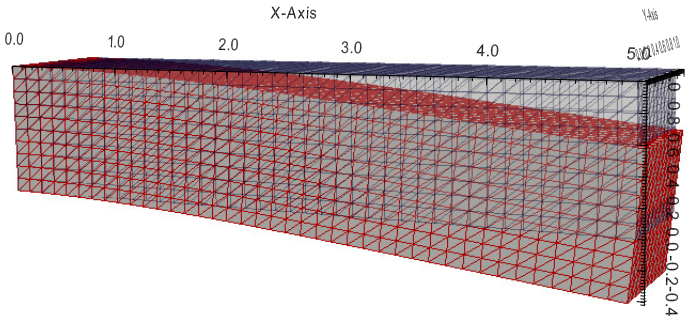


Figure 11: Result for the 3D elasticity problem

2.4.2 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 12).

The 2D problem to solve is the following

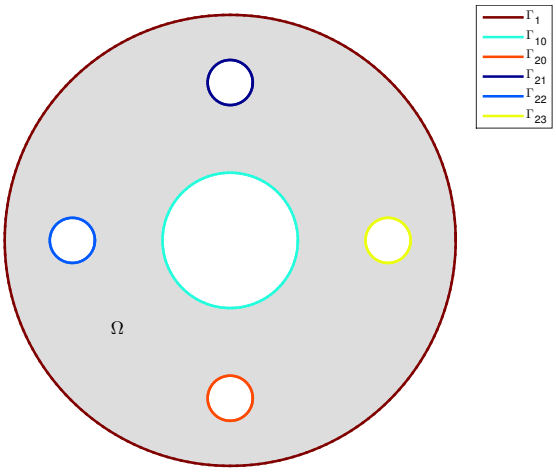


Figure 12: Domain and boundaries

2D problem : stationary heat with potential flow

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

$$
\begin{aligned}
-\operatorname{div}(\alpha \nabla u) + \langle \boldsymbol{V}, \nabla u \rangle + \beta u &= 0 \ \text{ in } \Omega \subset \mathbb{R}^2, && (2.39)\\
u &= 20 * y \ \text{ on } \Gamma_{21}, && (2.40)\\
u &= 0 \ \text{ on } \Gamma_{22} \cup \Gamma_{23}, && (2.41)\\
\frac{\partial u}{\partial n} &= 0 \ \text{ on } \Gamma_1 \cup \Gamma_{10} \cup \Gamma_{20} && (2.42)
\end{aligned}
$$

where Ω and its boundaries are given in Figure 12. This problem is well posed if $\alpha(\boldsymbol{x}) > 0$ and $\beta(\boldsymbol{x}) \geqslant 0$.
We choose α and β in Ω as :

$$
\begin{aligned}
\alpha(\boldsymbol{x}) &= 0.1 + x_2^2,\\
\beta(\boldsymbol{x}) &= 0.01
\end{aligned}
$$

The potential flow is the velocity field $\boldsymbol{V} = \nabla \phi$ where the scalar function ϕ is the velocity potential solution of the BVP

Velocity potential in 2D

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

$$
\begin{aligned}
-\Delta \phi &= 0 \ \text{ in } \Omega, && (2.43)\\
\phi &= -20 \ \text{ on } \Gamma_{21}, && (2.44)\\
\phi &= 20 \ \text{ on } \Gamma_{20}, && (2.45)\\
\frac{\partial \phi}{\partial n} &= 0 \ \text{ on } \Gamma_1 \cup \Gamma_{23} \cup \Gamma_{22} && (2.46)
\end{aligned}
$$

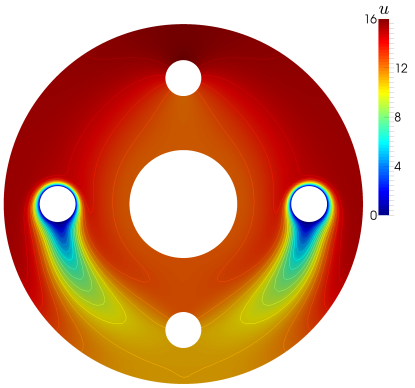
Then the potential flow $\boldsymbol{V}$ is solution of

Potential flow in 2D

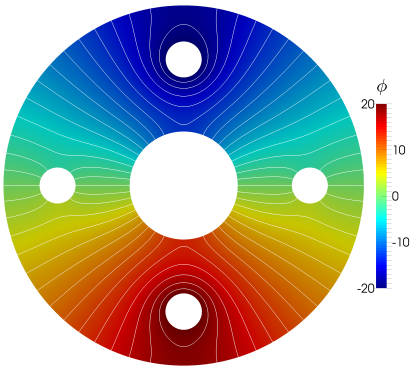
Find $\boldsymbol{V} = (\boldsymbol{V}_1, \boldsymbol{V}_2) \in \mathrm{H}^1(\Omega) \times \mathrm{H}^1(\Omega)$ such that

$$\boldsymbol{V} = \nabla \phi \text{ in } \Omega, \tag{2.47}$$

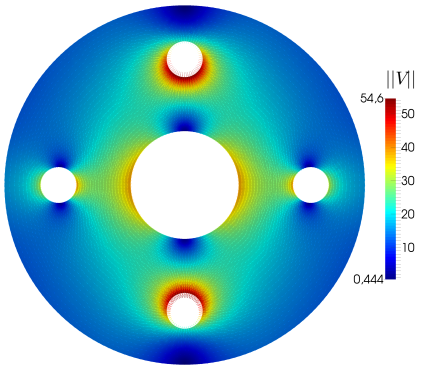
For a given mesh, the numerical result for heat u is represented in Figure 13a, velocity potential ϕ and potential flow $\boldsymbol{V}$ are shown on Figure 13.



(a) heat u



(b) Velocity potential ϕ



(c) Potential flow $\boldsymbol{V}$

Figure 13: Stationary heat with potential flow in 2D

Now we will present two manners of solving these problems using `vecFEMP1` codes.

Method 1 : split in three parts

The 2D potential velocity problem (2.43)-(2.46) can be equivalently expressed as the scalar BVP (1.2)-(1.4) :

We present now to

2D potential velocity as a *scalar* BVP

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

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

where

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

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

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

The algorithm using the toolbox for solving (2.43)-(2.46) is the following:

Algorithm 2.10 Velocity Potential in 2D

```
1: Dop ← LOPERATOR(𝕀, 0, 0, 0)                    ▷ Stiffness operator
2: pde ← INITPDE(Dop, 𝒯_h)
3: pde ← SETBCLABEL(pde, 20, 1, 'Dirichlet', 20)   ▷ u = 20 on Γ_20
4: pde ← SETBCLABEL(pde, 21, 1, 'Dirichlet', −20)  ▷ u = −20 on Γ_21
5: x_φ ← SOLVEPDE(pde)
```

Now to compute $\boldsymbol{V}$, we can write the potential flow problem (2.47) with $\mathcal{H}$-operators as

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

where

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

The algorithm using the toolbox for solving this problem is the following:

Algorithm 2.11 Potential flow in 2D

1: $\texttt{Aop} \leftarrow \textsc{InitHoperator}(2,2)$
2: $\texttt{Aop}.H(1,1) \leftarrow \textsc{Loperator}(2,\mathbb{O}_2,\mathbf{0},\mathbf{0},1)$
3: $\texttt{Aop}.H(2,2) \leftarrow \textsc{Loperator}(2,\mathbb{O}_2,\mathbf{0},\mathbf{0},1)$
4: $\texttt{Bop} \leftarrow \textsc{InitHoperator}(2,2)$
5: $\texttt{Bop}.H(1,1) \leftarrow \textsc{Loperator}(2,\mathbb{O}_2,\mathbf{0},\begin{pmatrix}1\\0\end{pmatrix},0)$
6: $\texttt{Bop}.H(2,2) \leftarrow \textsc{Loperator}(2,\mathbb{O}_2,\mathbf{0},\begin{pmatrix}0\\1\end{pmatrix},0)$
7: $\mathbb{A} \leftarrow \textsc{HAssemblyP1\_OptV3}(\texttt{Aop},\mathcal{T}_h)$
8: $\mathbb{B} \leftarrow \textsc{HAssemblyP1\_OptV3}(\texttt{Bop},\mathcal{T}_h)$
9: $\boldsymbol{b} \leftarrow \mathbb{B}\begin{pmatrix}\boldsymbol{x}_\phi\\\boldsymbol{x}_\phi\end{pmatrix}$
10: $\boldsymbol{V} \leftarrow \textsc{Solve}(\mathbb{A},\boldsymbol{b})$ ▷ Solve the linear system $\mathbb{A}\boldsymbol{V}=\boldsymbol{b}$

Finally, the stationary heat BVP (2.39)-(2.42) can be equivalently expressed as the scalar BVP (1.2)-(1.4) :

2D stationary heat as a *scalar* BVP

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

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

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

The algorithm using the toolbox for solving (2.39)-(2.42) is the following:

Algorithm 2.12 Stationary heat in 2D

1: $\texttt{Lop} \leftarrow \textsc{Loperator}(\begin{pmatrix}\alpha & 0\\ 0 & \alpha\end{pmatrix},\mathbf{0},\boldsymbol{V},\beta)$
2: pde $\leftarrow \textsc{initPDE}(\texttt{Lop},\mathcal{T}_h)$
3: pde $\leftarrow \textsc{setBCLabel}(\text{pde},21,1,\texttt{'Dirichlet'},\boldsymbol{x}\mapsto 20\boldsymbol{x}_2)$
4: pde $\leftarrow \textsc{setBCLabel}(\text{pde},22,1,\texttt{'Dirichlet'},0)$
5: pde $\leftarrow \textsc{setBCLabel}(\text{pde},23,1,\texttt{'Dirichlet'},0)$
6: $\boldsymbol{u} \leftarrow \textsc{SolvePDE}(\text{pde})$

We give respectively in Listing 11 and 12 the corresponding Matlab/Octave and Python codes.

Listing 11: **Stationary heat with potential flow in 2D, Matlab/Octave code (method 1)**

```
1
2  fprintf('1. Reading of the mesh\n');
3  Th=GetMesh2DOpt('FlowVelocity2D01-50.msh');
4  fprintf('2.a) Definition of a 2D velocity potential BVP\n');
5  Lop=Loperator(Th.d,{1,[];[],1},[],[],[]);
6  pde=initPDE(Lop,Th);
7  pde=setBC_PDE(pde,20,1,'Dirichlet',20);
8  pde=setBC_PDE(pde,21,1,'Dirichlet',-20);
9  fprintf('2.b) Solving 2D velocity potential BVP\n')
10 phi=solvePDE(pde);
11 fprintf('3) Setting/Solving 2D velocity field problem\n');
12 m=2;
13 Hop=Hoperator(d,m);
14 Hop.H{1,1}=Loperator(d,[],[],[],1);
15 Hop.H{2,2}=Loperator(d,[],[],[],1);
16 Bop=Hoperator(d,m);
17 Bop.H{1,1}=Loperator(d,[],[],{1;0},[]);
18 Bop.H{2,2}=Loperator(d,[],[],{0;1},[]);
19 A=HAssemblyP1_OptV3(Th,Hop);
20 B=HAssemblyP1_OptV3(Th,Bop);
21 U=A\(B*[phi;phi]);
22 V=splitSol(U,2,Th.nq);
23 fprintf('4.a) Definition of a 2D stationary heat BVP with potential flow\n');
24 af=@(x,y) 0.1+y.^2;
25 Dop=Loperator(Th.d,{af,[];[],af},[],{V{1},V{2}},0.01);
26 pdeHeat=initPDE(Dop,Th);
27 pdeHeat=setBC_PDE(pdeHeat,21,1,'Dirichlet', @(x,y) 20*y   );
28 pdeHeat=setBC_PDE(pdeHeat,22,1,'Dirichlet', 0   );
29 pdeHeat=setBC_PDE(pdeHeat,23,1,'Dirichlet', 0   );
30 fprintf('4.b) Solving 2D stationary heat BVP with potential flow\n');
31 u=solvePDE(pdeHeat);
```

Listing 12: **Stationary heat with potential flow in 2D, Python code (method 1)**

```
1  d=2
2  print('1. Reading of the mesh')
3  Th=readFreeFEM("FlowVelocity2D01-50.msh")
4  print("2.a) Definition of a 2D velocity potential BVP")
5  Lop=Loperator(d=Th.d,A=[[1,None],[None,1]])
6  pde=initPDE(Lop,Th);
7  pde=setBC_PDE(pde,20,1,"Dirichlet",20,None);
8  pde=setBC_PDE(pde,21,1,"Dirichlet",-20,None);
9  print("2.b) Solving 2D velocity potential BVP")
10 phi=solvePDE(pde)
11 print("3. Setting/Solving 2D velocity field problem")
12 Hop=Hoperator(d=2,m=2)
13 Hop.H[0][0]=Loperator(d=d,a0=1)
14 Hop.H[1][1]=Loperator(d=d,a0=1)
15 Bop=Hoperator(d=2,m=2)
16 Bop.H[0][0]=Loperator(d=d,c=[1,0])
17 Bop.H[1][1]=Loperator(d=d,c=[0,1])
18 A=HAssemblyP1_OptV3(Th,Hop,1)
19 B=HAssemblyP1_OptV3(Th,Bop,1)
20 b=B*np.hstack([phi,phi])
21 U=spsolve(A,b)
22 V=splitSol(U,2,Th.nq)
23 print('4.a) Definition of a 2D stationary heat BVP with potential flow')
24 af=lambda x,y: 0.1+y**2;
25 Lop=Loperator(d=Th.d,A=[[af,None],[None,af]],c=[V[0],V[1]],a0=0.01);
26 pdeHeat=initPDE(Lop,Th)
27 pdeHeat=setBC_PDE(pdeHeat,21,0,'Dirichlet', lambda x,y: 20*y )
28 pdeHeat=setBC_PDE(pdeHeat,22,0,'Dirichlet', 0)
29 pdeHeat=setBC_PDE(pdeHeat,23,0,'Dirichlet', 0)
30 print('4.b) Solving 2D stationary heat PDE with potential flow')
31 u=solvePDE(pdeHeat)
```

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

We can merged velocity potential BVP (2.43)-(2.46) and potential flow to obtain the new BVP

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

$$
\begin{aligned}
-\left(\frac{\partial \boldsymbol{V}_1}{\partial x} + \frac{\partial \boldsymbol{V}_2}{\partial y}\right) &= 0 \ \text{ in } \Omega, && (2.48)\\
\boldsymbol{V}_1 - \frac{\partial \phi}{\partial x} &= 0 \ \text{ in } \Omega, && (2.49)\\
\boldsymbol{V}_2 - \frac{\partial \phi}{\partial y} &= 0 \ \text{ in } \Omega, && (2.50)\\
\phi &= -20 \ \text{ on } \Gamma_{21}, && (2.51)\\
\phi &= 20 \ \text{ on } \Gamma_{20}, && (2.52)\\
\frac{\partial \phi}{\partial n} &= 0 \ \text{ on } \Gamma_1 \cup \Gamma_{23} \cup \Gamma_{22} && (2.53)
\end{aligned}
$$

We can also replace (2.48) by $-\Delta\phi = 0$.

Let $\boldsymbol{w} = \begin{pmatrix} \phi \\ \boldsymbol{V}_1 \\ \boldsymbol{V}_2 \end{pmatrix}$, the previous problem (2.48)-(2.53) can be equivalently expressed as the vector BVP (1.10)-(1.12) :

Vector BVP

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

$$
\begin{aligned}
\mathcal{H}(\boldsymbol{w}) &= \boldsymbol{f} && \text{in } \Omega, && (2.54)\\
\boldsymbol{w}_\alpha &= g_\alpha^D && \text{on } \Gamma_\alpha^D, \ \forall \alpha \in [\![1,3]\!], && (2.55)\\
\frac{\partial \boldsymbol{w}}{\partial n_{\mathcal{H}_\alpha}} + a_\alpha^R \boldsymbol{w}_\alpha &= g_\alpha^R && \text{on } \Gamma_\alpha^R, \ \forall \alpha \in [\![1,3]\!], && (2.56)
\end{aligned}
$$

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

$$
\begin{aligned}
&\frac{\partial \boldsymbol{w}_1}{\partial n_{\mathcal{H}_{1,1}}} = 0, && \frac{\partial \boldsymbol{w}_2}{\partial n_{\mathcal{H}_{1,2}}} = \boldsymbol{w}_2 \boldsymbol{n}_1, && \frac{\partial \boldsymbol{w}_3}{\partial n_{\mathcal{H}_{1,3}}} = \boldsymbol{w}_3 \boldsymbol{n}_2,\\
&\frac{\partial \boldsymbol{w}_1}{\partial n_{\mathcal{H}_{2,1}}} = 0, && \frac{\partial \boldsymbol{w}_2}{\partial n_{\mathcal{H}_{2,2}}} = 0, && \frac{\partial \boldsymbol{w}_3}{\partial n_{\mathcal{H}_{2,3}}} = 0\\
&\frac{\partial \boldsymbol{w}_1}{\partial n_{\mathcal{H}_{3,1}}} = 0, && \frac{\partial \boldsymbol{w}_2}{\partial n_{\mathcal{H}_{3,2}}} = 0, && \frac{\partial \boldsymbol{w}_3}{\partial n_{\mathcal{H}_{3,3}}} = 0.
\end{aligned}
$$

So we obtain

$$
\frac{\partial \boldsymbol{w}}{\partial n_{\mathcal{H}_1}} \overset{\text{def}}{=} \sum_{\alpha=1}^{3} \frac{\partial \boldsymbol{w}_\alpha}{\partial n_{\mathcal{H}_{1,\alpha}}} = \langle \boldsymbol{V}, \boldsymbol{n} \rangle = \frac{\partial \phi}{\partial \boldsymbol{n}}, \tag{2.57}
$$

and

$$\frac{\partial \boldsymbol{w}}{\partial n_{\mathcal{H}_2}} = \frac{\partial \boldsymbol{w}}{\partial n_{\mathcal{H}_3}} := 0. \tag{2.58}$$

From (2.58), 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 given on lines 3 to 13 of Algorithm 2.13.

Algorithm 2.13 Stationary heat with potential velocity problem (method 2)

1: $\mathcal{T}_h \leftarrow$ getMesh(...) ▷ Load FreeFEM++ mesh
2: $\boldsymbol{e}_1 \leftarrow \begin{pmatrix}1\\0\end{pmatrix}$, $\boldsymbol{e}_2 \leftarrow \begin{pmatrix}0\\1\end{pmatrix}$
3: Hop $\leftarrow$ Hoperator$(2,3)$
4: Hop.H$(1,2) \leftarrow$ Loperator$(\mathbb{O}_2, -\boldsymbol{e}_1, \boldsymbol{0}, 0)$
5: Hop.H$(1,3) \leftarrow$ Loperator$(\mathbb{O}_2, -\boldsymbol{e}_2, \boldsymbol{0}, 0)$
6: Hop.H$(2,1) \leftarrow$ Loperator$(\mathbb{O}_2, \boldsymbol{0}, -\boldsymbol{e}_1, 0)$
7: Hop.H$(2,2) \leftarrow$ Loperator$(\mathbb{O}_2, \boldsymbol{0}, \boldsymbol{0}, 1)$
8: Hop.H$(3,1) \leftarrow$ Loperator$(\mathbb{O}_2, \boldsymbol{0}, -\boldsymbol{e}_2, 0)$
9: Hop.H$(3,3) \leftarrow$ Loperator$(\mathbb{O}_2, \boldsymbol{0}, \boldsymbol{0}, 1)$
10: PDEflow $\leftarrow$ initPDE(Hop, $\mathcal{T}_h$)
11: PDEflow $\leftarrow$ setBC_PDE(PDEflow, 20, 1, 'Dirichlet', 20., $\varnothing$)
12: PDEflow $\leftarrow$ setBC_PDE(PDEflow, 21, 1, 'Dirichlet', $-20.$, $\varnothing$)
13: $[\boldsymbol{\phi}, \boldsymbol{V}_1, \boldsymbol{V}_2] \leftarrow$ SolvePDE(PDEflow)
14: $\alpha \leftarrow (x,y) \longmapsto 0.1 + y^2$
15: $g_{21} \leftarrow (x,y) \longmapsto 20y$
16: $\beta \leftarrow 0.01$
17: Dop $\leftarrow$ Loperator$\left(\begin{pmatrix}\alpha & 0\\ 0 & \alpha\end{pmatrix}, \boldsymbol{0}, \begin{pmatrix}\boldsymbol{V}_1\\ \boldsymbol{V}_2\end{pmatrix}, \beta\right)$
18: PDE $\leftarrow$ initPDE(Dop, $\mathcal{T}_h$) ▷ Set homogeneous 'Neumann' condition on all boundaries
19: PDE $\leftarrow$ setBC_PDE(PDE, 21, 1, 'Dirichlet', g_{21}, $\varnothing$) ▷ $u = 4$ on Γ_2
20: PDE $\leftarrow$ setBC_PDE(PDE, 22, 1, 'Dirichlet', 0, $\varnothing$) ▷ $u = -4$ on Γ_4
21: PDE $\leftarrow$ setBC_PDE(PDE, 23, 1, 'Dirichlet', 0, $\varnothing$ ▷ $u = 0$ on Γ_{20}
22: $\boldsymbol{u} \leftarrow$ SolvePDE(PDE)

We give respectively in Listing 13 and 14 the corresponding Matlab/Octave and Python codes.

Listing 13: **Stationary heat with potential flow in 2D, Matlab/Octave code (method 2)**

```
1
2  d=2;
3  fprintf('1. Reading of the mesh \n');
4  Th=GetMesh2DOpt('FlowVelocity2D01-50.msh');
5  fprintf('2. Setting 2D potential velocity/flow BVP\n');
6  Hop=Hoperator(d,3);
7  Hop.H{1,2}=Loperator(d,[],{-1,0},[],[]);
8  Hop.H{1,3}=Loperator(d,[],{0,-1},[],[]);
9  Hop.H{2,1}=Loperator(d,[],[],{-1,0},[]);
10 Hop.H{2,2}=Loperator(d,[],[],[],1);
11 Hop.H{3,1}=Loperator(d,[],[],{0,-1},[]);
12 Hop.H{3,3}=Loperator(d,[],[],[],1);
13 pdeFlow=initPDE(Hop,Th);
14 pdeFlow=setBC_PDE(pdeFlow,20,1,'Dirichlet',20);
15 pdeFlow=setBC_PDE(pdeFlow,21,1,'Dirichlet',-20);
16 fprintf('3. Solving 2D potential velocity/flow BVP\n')
17 U=solvePDE(pdeFlow,'split',true);
18 fprintf('4. Setting 2D stationary heat BVP with potential flow\n');
19 af=@(x,y) 0.1+y.^2;
20 Dop=Loperator(Th.d,{af,[];[],af},[],{U{2},U{3}},0.01);
21 pdeHeat=initPDE(Dop,Th);
22 pdeHeat=setBC_PDE(pdeHeat,21,1,'Dirichlet', @(x,y) 20*y   );
23 pdeHeat=setBC_PDE(pdeHeat,22,1,'Dirichlet', 0   );
24 pdeHeat=setBC_PDE(pdeHeat,23,1,'Dirichlet', 0   );
25 fprintf('5. Solving 2D stationary heat BVP with potential flow\n');
26 x=solvePDE(pdeHeat);
```

Listing 14: **Stationary heat with potential flow in 2D, Python code (method 2)**

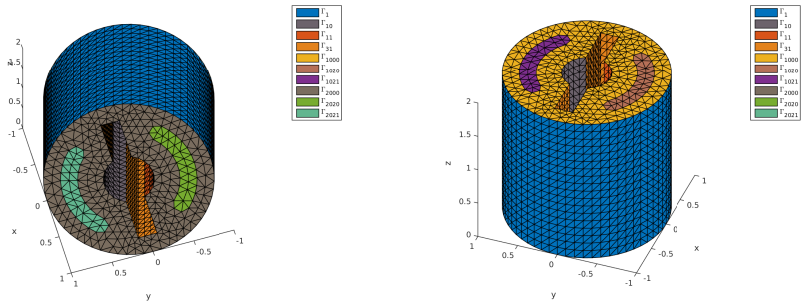
```
1
2  d=2;m=3;
3  print('1. Reading of the mesh')
4  Th=readFreeFEM("FlowVelocity2D01-50.msh")
5  print("2. Setting 2D potential velocity/flow BVP")
6  Hop=Hoperator(d=2,m=3)
7  #Hop.H[0][0]=Loperator(d=d,A=[[1,None],[None,1]])
8  Hop.H[0][1]=Loperator(d=d,b=[-1,None])
9  Hop.H[0][2]=Loperator(d=d,b=[None,-1])
10 Hop.H[1][0]=Loperator(d=d,c=[-1,None])
11 Hop.H[1][1]=Loperator(d=d,a0=1)
12 Hop.H[2][0]=Loperator(d=d,c=[None,-1])
13 Hop.H[2][2]=Loperator(d=d,a0=1)
14 pdeFlow=initPDE(Hop,Th);
15 pdeFlow=setBC_PDE(pdeFlow,20,1,"Dirichlet",20,None);
16 pdeFlow=setBC_PDE(pdeFlow,21,1,"Dirichlet",-20,None);
17 print("3. Solving 2D potential velocity/flow BVP")
18 U=solvePDE(pdeFlow,split=True)
19 print('4. Setting 2D stationary heat BVP with potential flow')
20 af=lambda x,y: 0.1+y**2;
21 Lop=Loperator(d=Th.d,A=[[af,None],[None,af]],c=[U[1],U[2]],a0=0.01);
22 pdeHeat=initPDE(Lop,Th)
23 pdeHeat=setBC_PDE(pdeHeat,21,0,'Dirichlet', lambda x,y: 20*y )
24 pdeHeat=setBC_PDE(pdeHeat,22,0,'Dirichlet', 0)
25 pdeHeat=setBC_PDE(pdeHeat,23,0,'Dirichlet', 0)
26 print('5. Solving 2D stationary heat PDE with potential flow')
27 u=solvePDE(pdeHeat)
```

2.4.3 Stationary heat with potential flow in 3D

Let $\Omega \subset \mathbb{R}^3$ be the cylinder given in Figure 14.

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



(a) Bottom face (b) Top face

Figure 14: Stationary heat with potential flow : 3d mesh

3D problem : stationary heat with potential flow

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

$$
\begin{aligned}
-\operatorname{div}(\alpha \nabla u) + \langle \boldsymbol{V}, \nabla u \rangle + \beta u &= 0 \ \text{ in } \Omega \subset \mathbb{R}^3, && (2.59)\\
u &= 30 \ \text{ on } \Gamma_{1020} \cup \Gamma_{2020}, && (2.60)\\
u &= 10\delta_{|z-1|>0.5} \ \text{ on } \Gamma_{10}, && (2.61)\\
\frac{\partial u}{\partial n} &= 0 \ \text{ otherwise} && (2.62)
\end{aligned}
$$

where Ω and its boundaries are given in Figure 14. This problem is well posed if $\alpha(\boldsymbol{x}) > 0$ and $\beta(\boldsymbol{x}) \geqslant 0$.
We choose α and β in Ω as :

$$
\begin{aligned}
\alpha(\boldsymbol{x}) &= 1,\\
\beta(\boldsymbol{x}) &= 0.01
\end{aligned}
$$

The potential flow is the velocity field $\boldsymbol{V} = \nabla \phi$ where the scalar function ϕ is the velocity potential solution of the PDE :

Velocity potential in 3d

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

$$
\begin{aligned}
-\Delta \phi &= 0 \ \text{ in } \Omega, && (2.63)\\
\phi &= 1 \ \text{ on } \Gamma_{1021} \cup \Gamma_{2021}, && (2.64)\\
\phi &= -1 \ \text{ on } \Gamma_{1020} \cup \Gamma_{2020}, && (2.65)\\
\frac{\partial \phi}{\partial n} &= 0 \ \text{ otherwise} && (2.66)
\end{aligned}
$$

To solve problem (2.59)-(2.62), we need to compute the velocity field $\boldsymbol{V}$. For that we can rewrite the potential flow problem (2.63)-(2.66), by introducing $\boldsymbol{V} = (\boldsymbol{V}_1, \boldsymbol{V}_2, \boldsymbol{V}_3)$ as unknowns :

Velocity potential and velocity field in 3d

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

$$-\left(\frac{\partial \boldsymbol{V}_1}{\partial x} + \frac{\partial \boldsymbol{V}_2}{\partial y} + \frac{\partial \boldsymbol{V}_3}{\partial z}\right) = 0 \text{ in } \Omega, \tag{2.67}$$

$$\boldsymbol{V}_1 - \frac{\partial \phi}{\partial x} = 0 \text{ in } \Omega, \tag{2.68}$$

$$\boldsymbol{V}_2 - \frac{\partial \phi}{\partial y} = 0 \text{ in } \Omega, \tag{2.69}$$

$$\boldsymbol{V}_3 - \frac{\partial \phi}{\partial z} = 0 \text{ in } \Omega, \tag{2.70}$$

with boundary conditions (2.64) to (2.66).

We can also replace (2.67) 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

$$\mathcal{H}(\boldsymbol{w}) = 0 \tag{2.71}$$

with

$$\mathcal{H}_{1,1} = 0, \quad \mathcal{H}_{1,2} = \mathcal{L}_{\mathbb{O},-\boldsymbol{e}_1,\boldsymbol{0},0}, \quad \mathcal{H}_{1,3} = \mathcal{L}_{\mathbb{O},-\boldsymbol{e}_2,\boldsymbol{0},0}, \quad \mathcal{H}_{1,4} = \mathcal{L}_{\mathbb{O},-\boldsymbol{e}_3,\boldsymbol{0},0}, \tag{2.72}$$

$$\mathcal{H}_{2,1} = \mathcal{L}_{\mathbb{O},\boldsymbol{0},-\boldsymbol{e}_1,0}, \quad \mathcal{H}_{2,2} = \mathcal{L}_{\mathbb{O},\boldsymbol{0},\boldsymbol{0},1}, \quad \mathcal{H}_{2,3} = 0, \quad \mathcal{H}_{2,4} = 0, \tag{2.73}$$

$$\mathcal{H}_{3,1} = \mathcal{L}_{\mathbb{O},\boldsymbol{0},-\boldsymbol{e}_2,0}, \quad \mathcal{H}_{3,2} = 0, \quad \mathcal{H}_{3,3} = \mathcal{L}_{\mathbb{O},\boldsymbol{0},\boldsymbol{0},1}, \quad \mathcal{H}_{3,4} = 0, \tag{2.74}$$

$$\mathcal{H}_{4,1} = \mathcal{L}_{\mathbb{O},\boldsymbol{0},-\boldsymbol{e}_3,0}, \quad \mathcal{H}_{4,2} = 0, \quad \mathcal{H}_{4,3} = 0, \quad \mathcal{H}_{4,4} = \mathcal{L}_{\mathbb{O},\boldsymbol{0},\boldsymbol{0},1}, \tag{2.75}$$

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

$$\sum_{\alpha=1}^{4} \frac{\partial \boldsymbol{w}_\alpha}{\partial n_{\mathcal{H}_{1,\alpha}}} = \langle \boldsymbol{V}, \boldsymbol{n} \rangle = \langle \nabla\,\phi, \boldsymbol{n} \rangle, \tag{2.76}$$

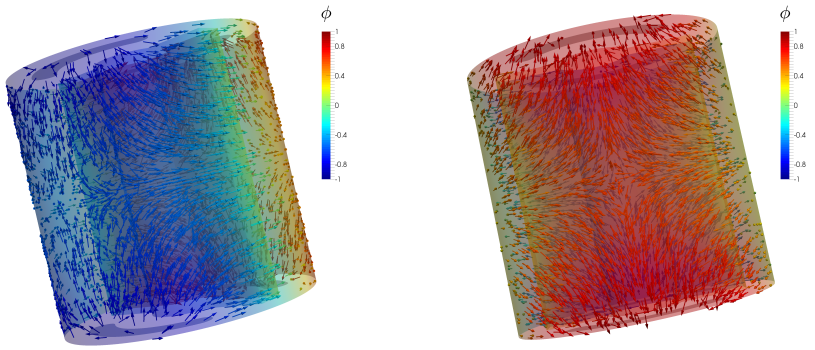
and

$$\sum_{\alpha=1}^{4} \frac{\partial \boldsymbol{w}_\alpha}{\partial n_{\mathcal{H}_{2,\alpha}}} = \sum_{\alpha=1}^{4} \frac{\partial \boldsymbol{w}_\alpha}{\partial n_{\mathcal{H}_{3,\alpha}}} = \sum_{\alpha=1}^{4} \frac{\partial \boldsymbol{w}_\alpha}{\partial n_{\mathcal{H}_{4,\alpha}}} = 0. \tag{2.77}$$

From (2.77), 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 (2.64) to (2.66), 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 solution of this vector boundary value problem is given in lines 3 to 13 of Algorithm 2.14. A representation of velocity potential ϕ and potential flow $\boldsymbol{V}$ is given in Figure 15.



(a) ϕ and $\boldsymbol{V}$: first view (b) ϕ and $\boldsymbol{V}$: second view

Figure 15: HeatAndFlowVelocity3d01 problem

The operator in (2.59) is given by $\mathcal{L}_{\alpha \mathbb{I}, \boldsymbol{0}, \boldsymbol{V}, \beta}$. The conormal derivative $\frac{\partial u}{\partial n_{\mathcal{L}}}$ is

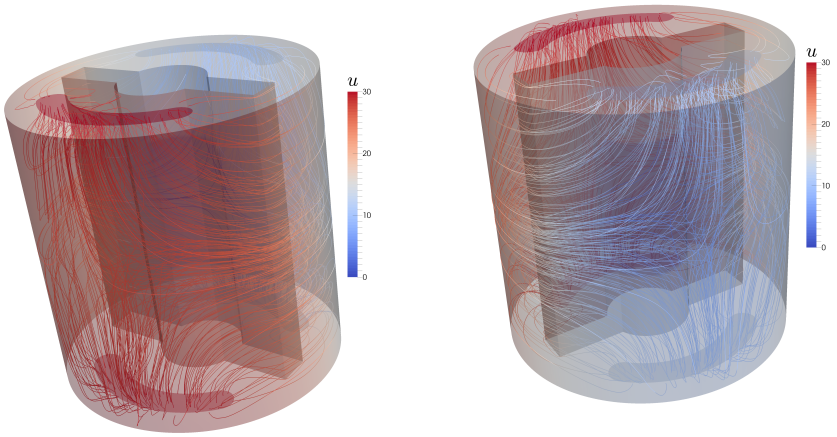
$$\frac{\partial u}{\partial n_{\mathcal{L}}} := \langle \mathbb{A} \nabla u, \boldsymbol{n} \rangle - \langle \boldsymbol{b} u, \boldsymbol{n} \rangle = \alpha \frac{\partial u}{\partial n}.$$

The algorithm using the toolbox for solving (2.67)-(2.70) is the following:

Algorithm 2.14 Stationary heat with potential velocity problem

1: $\mathcal{T}_h \leftarrow$ getMesh(...) ▷ Load FreeFEM++ mesh
2: $\boldsymbol{e}_1 \leftarrow \begin{pmatrix}1\\0\\0\end{pmatrix}$, $\boldsymbol{e}_2 \leftarrow \begin{pmatrix}0\\1\\0\end{pmatrix}$, $\boldsymbol{e}_3 \leftarrow \begin{pmatrix}0\\0\\1\end{pmatrix}$
3: Hop $\leftarrow$ Hoperator$(3,4)$
4: Hop.H$(1,2) \leftarrow$ Loperator$(\mathbb{O}_3, -\boldsymbol{e}_1, \mathbf{0}, 0)$
5: Hop.H$(1,3) \leftarrow$ Loperator$(\mathbb{O}_3, -\boldsymbol{e}_2, \mathbf{0}, 0)$
6: Hop.H$(1,4) \leftarrow$ Loperator$(\mathbb{O}_3, -\boldsymbol{e}_3, \mathbf{0}, 0)$
7: Hop.H$(2,1) \leftarrow$ Loperator$(\mathbb{O}_3, \mathbf{0}, -\boldsymbol{e}_1, 0)$, Hop.H$(2,2) \leftarrow$ Loperator$(\mathbb{O}_3, \mathbf{0}, \mathbf{0}, 1)$
8: Hop.H$(3,1) \leftarrow$ Loperator$(\mathbb{O}_3, \mathbf{0}, -\boldsymbol{e}_2, 0)$, Hop.H$(3,3) \leftarrow$ Loperator$(\mathbb{O}_3, \mathbf{0}, \mathbf{0}, 1)$
9: Hop.H$(4,1) \leftarrow$ Loperator$(\mathbb{O}_3, \mathbf{0}, -\boldsymbol{e}_3, 0)$, Hop.H$(4,4) \leftarrow$ Loperator$(\mathbb{O}_3, \mathbf{0}, \mathbf{0}, 1)$
10: PDEflow $\leftarrow$ initPDE(Hop, $\mathcal{T}_h$)
11: PDEflow $\leftarrow$ setBC_PDE(PDEflow, 20, 1, 'Dirichlet', 20., $\varnothing$)
12: PDEflow $\leftarrow$ setBC_PDE(PDEflow, 21, 1, 'Dirichlet', $-20.$, $\varnothing$)
13: $[\boldsymbol{\phi}, \boldsymbol{V}_1, \boldsymbol{V}_2, \boldsymbol{V}_3] \leftarrow$ SolvePDE(PDEflow)
14: $\alpha \leftarrow (x,y,z) \longmapsto 1$
15: $g_{20} \leftarrow (x,y,z) \longmapsto 30$, $g_{10} \leftarrow (x,y,z) \longmapsto 10 * (|z-1| > 0.5)$
16: $\beta \leftarrow 0.01$
17: Dop $\leftarrow$ Loperator$\left(\begin{pmatrix}\alpha & 0 & 0\\ 0 & \alpha & 0\\ 0 & 0 & \alpha\end{pmatrix}, \mathbf{0}, \begin{pmatrix}\boldsymbol{V}_1\\ \boldsymbol{V}_2\\ \boldsymbol{V}_3\end{pmatrix}, \beta\right)$
18: PDE $\leftarrow$ initPDE(Dop, $\mathcal{T}_h$) ▷ Set homogeneous 'Neumann' condition on all boundaries
19: PDE $\leftarrow$ setBC_PDE(PDE, 1020, 1, 'Dirichlet', g_{20}, $\varnothing$)
20: PDE $\leftarrow$ setBC_PDE(PDE, 2022, 1, 'Dirichlet', g_{20}, $\varnothing$)
21: PDE $\leftarrow$ setBC_PDE(PDE, 10, 1, 'Dirichlet', g_{10}, $\varnothing$)
22: $\boldsymbol{u} \leftarrow$ SolvePDE(PDE)

The numerical solution for a given mesh is shown on Figure 16



(a) us solution with streamline : first viewview

(b) us solution with streamline : second

Figure 16: HeatAndFlowVelocity3d01 problem

2.4.4 Biharmonic problems

The biharmonic equation is the fourth-order partial PDE given by

$$\Delta^2 u = f \tag{2.78}$$

where $\Delta^2 u = \Delta(\Delta u) = \sum_{i=1}^{d}\sum_{j=1}^{d} \frac{\partial^4 u}{\partial x_i^2 \partial x_j^2}$

The boundary conditions on Γ can be

- *Clamped Plate* (CP) or *pure Dirichlet* type:

$$u = \frac{\partial u}{\partial \boldsymbol{n}} = g \tag{2.79}$$

- *Simply Supported Plate* (SSP) or *Navier* type :

$$u = \Delta u = g \tag{2.80}$$

- *Pure Hinged Plate* or *Stelov* type :

$$u = \Delta u - (1-\sigma) K \frac{\partial u}{\partial \boldsymbol{n}} = g \tag{2.81}$$

- *Cahn-Hilliard* (CH) type

$$\frac{\partial u}{\partial n} = \frac{\partial \Delta u}{\partial n} = g \tag{2.82}$$

Link with $\mathcal{H}$-operator and boundary conditions

Classically the fourth-order PDE (2.78) is converted to the two second-order PDE

$$-\Delta u = v \tag{2.83}$$
$$-\Delta v = f \tag{2.84}$$

These two equations can be equivalently written as

$$\mathcal{G}\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} f \\ 0 \end{pmatrix} \quad \text{or} \quad \mathcal{K}\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} 0 \\ f \end{pmatrix} \tag{2.85}$$

where $\mathcal{G}$ and $\mathcal{K}$ are the $\mathcal{H}$-operators defined by

$$\mathcal{G} = \begin{pmatrix} 0 & \mathcal{L}_{\mathbb{I},\mathbf{0},\mathbf{0},0} \\ \mathcal{L}_{\mathbb{I},\mathbf{0},\mathbf{0},0} & \mathcal{L}_{\mathbb{O},\mathbf{0},\mathbf{0},-1} \end{pmatrix} \quad \text{and} \quad \mathcal{K} = \begin{pmatrix} \mathcal{L}_{\mathbb{I},\mathbf{0},\mathbf{0},0} & \mathcal{L}_{\mathbb{O},\mathbf{0},\mathbf{0},-1} \\ \mathcal{L}_{\mathbb{O},\mathbf{0},\mathbf{0},0} & \mathcal{L}_{\mathbb{I},\mathbf{0},\mathbf{0},0} \end{pmatrix} \tag{2.86}$$

Let $\boldsymbol{w} = (u, v)$. From (2.85), the components of the conormal derivative of $\boldsymbol{w}$ defined in (1.13) are given by

$$\begin{aligned} \frac{\partial \boldsymbol{w}}{\partial n_{\mathcal{K}_1}} &\overset{\text{def}}{=} \sum_{\beta=1}^{2} \frac{\partial \boldsymbol{w}_\beta}{\partial n_{\mathcal{K}_{1,\beta}}} = \sum_{\beta=1}^{2} \left\langle \mathbb{A}^{1,\beta} \nabla \boldsymbol{w}_\beta, \boldsymbol{n} \right\rangle - \left\langle \boldsymbol{b}^{1,\beta} \boldsymbol{u}_\beta, \boldsymbol{n} \right\rangle \\ &= \langle \mathbb{I} \nabla \boldsymbol{w}_1, \boldsymbol{n} \rangle = \langle \nabla u, \boldsymbol{n} \rangle \\ &= \frac{\partial u}{\partial \boldsymbol{n}} = \frac{\partial \boldsymbol{w}}{\partial n_{\mathcal{G}_2}} \end{aligned} \tag{2.87}$$

and

$$\begin{aligned}\frac{\partial \boldsymbol{w}}{\partial n_{\mathcal{K}_2}} &\overset{\text{def}}{=} \sum_{\beta=1}^{2} \frac{\partial \boldsymbol{w}_\beta}{\partial n_{\mathcal{K}_{2,\beta}}} = \sum_{\beta=1}^{2} \left\langle \mathbb{A}^{2,\beta} \nabla \boldsymbol{w}_\beta, \boldsymbol{n} \right\rangle - \left\langle \boldsymbol{b}^{2,\beta} \boldsymbol{u}_\beta, \boldsymbol{n} \right\rangle \\ &= \left\langle \mathbb{I} \nabla \boldsymbol{w}_2, \boldsymbol{n} \right\rangle = \left\langle \nabla v, \boldsymbol{n} \right\rangle \\ &= \frac{\partial v}{\partial \boldsymbol{n}} = \frac{\partial \boldsymbol{w}}{\partial n_{\mathcal{G}_1}} \end{aligned} \tag{2.88}$$

Clamped plate problem

In this part, we take examples of the thesis of T. Gerasimov [2] (page 138).

Let $d = 2$, $\Omega = [-1, 6] \times [-1, 1] \subset \mathbb{R}^d$ and

$$f := \exp(-100((x + 0.75)^2 + (y - 0.75)^2)).$$

Clamped plate problem

Find u such that

$$\begin{cases} \Delta^2 u &= f, \quad \text{in } \Omega \subset \mathbb{R}^d, \ d = 2 \\ u &= 0, \quad \text{on } \Gamma \\ \dfrac{\partial u}{\partial \boldsymbol{n}} &= 0. \quad \text{on } \Gamma \end{cases} \tag{2.89}$$

Let $v = -\Delta u$. Then the problem (2.89) can be equivalently written as the split problem (2.90).

Clamped plate split problem

Find u and v such that

$$\begin{cases} v &= -\Delta u, \quad \text{in } \Omega \\ -\Delta v &= f, \quad \text{in } \Omega \\ u &= 0, \quad \text{on } \Gamma \\ \dfrac{\partial u}{\partial \boldsymbol{n}} &= 0. \quad \text{on } \Gamma \end{cases} \tag{2.90}$$

Using the operator $\mathcal{G}$ defined in (2.86) and its conormal derivatives (2.87)-(2.88), we can write the vector BVP associated with (2.90) as

***Vector* BVP for clamped plate problem (2.90) with $\mathcal{G}$ operator**

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

$$\begin{aligned} \mathcal{G}(\boldsymbol{w}) &= \begin{pmatrix} f \\ 0 \end{pmatrix} && \text{in } \Omega, \\ \boldsymbol{w}_1 &= 0 && \text{on } \Gamma_1^D = \Gamma \ (\text{so } \ \Gamma_1^R = \varnothing \ ) \\ \frac{\partial \boldsymbol{w}}{\partial n_{\mathcal{G}_2}} &= 0 && \text{on } \Gamma_2^R = \Gamma \ (\text{so } \ \Gamma_2^D = \varnothing \ ) \end{aligned}$$

Remark 2. *We cannot use the operator* $\mathcal{K}$ *defined in* (2.86) *due to a boundary condition trouble. Indeed* $\dfrac{\partial \boldsymbol{w}}{\partial n_{\mathcal{K}_1}} = \dfrac{\partial u}{\partial n}$ *and we cannot set a Dirichlet condition* $\boldsymbol{w}_1 = 0$ *on* $\Gamma_1^D = \Gamma$ *with a Neumann condition* $\dfrac{\partial \boldsymbol{w}}{\partial n_{\mathcal{K}_1}} = 0$ *on* $\Gamma_1^R = \Gamma$.

The domain Ω could be generated with the HyperCube function :

$$\mathcal{T}_h \leftarrow \text{HyperCube}(2, [70, 20], \boldsymbol{x} \mapsto (-1 + 7\boldsymbol{x}_1, -1 + 2\boldsymbol{x}_2)).$$

In Figure 17 we represent Ω and its boundary $\Gamma = \Gamma_1 \cup \Gamma_2 \cup \Gamma_3 \cup \Gamma_4$.

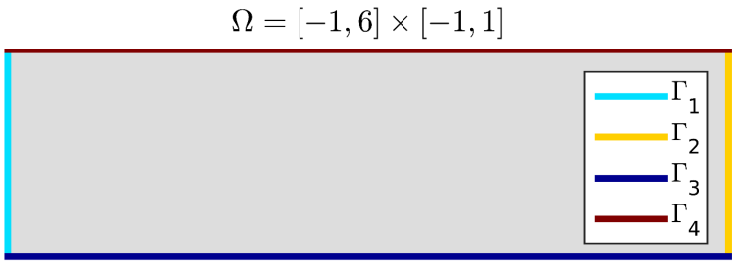


Figure 17: Mesh from $\text{HyperCube}(2, [70, 20], \boldsymbol{x} \mapsto (-1 + 7\boldsymbol{x}_1, -1 + 2\boldsymbol{x}_2))$

The algorithm using the toolbox to solve this vector BVP is the following:

Algorithm 2.15 Clamped plate problem

$\mathcal{T}_h \leftarrow \text{HyperCube}(2, [70, 20], \boldsymbol{x} \mapsto (-1 + 7\boldsymbol{x}_1, -1 + 2\boldsymbol{x}_2))$
$\mathtt{d} \leftarrow 2,\ \mathtt{m} \leftarrow 2$
$\mathtt{Hop} \leftarrow \text{Hoperator}(2, 2)$
$\mathtt{Hop.H}(1, 2) \leftarrow \text{Loperator}(2, \mathbb{I}_2, \mathbf{0}, \mathbf{0}, 0)$
$\mathtt{Hop.H}(2, 1) \leftarrow \text{Loperator}(2, \mathbb{I}_2, \mathbf{0}, \mathbf{0}, 0)$
$\mathtt{Hop.H}(2, 2) \leftarrow \text{Loperator}(2, \mathbb{O}_2, \mathbf{0}, \mathbf{0}, -1)$
pde $\leftarrow \text{initPDE}(\mathtt{Hop}, \mathcal{T}_h)$
pde.f $\leftarrow \boldsymbol{x} \mapsto \begin{pmatrix} \exp(-100((\boldsymbol{x}_1 + 0.75)^2 + (\boldsymbol{x}_2 - 0.75)^2)) \\ 0 \end{pmatrix}$
for $i \leftarrow 1$ **to** pde.nlab **do**
  pde $\leftarrow \text{setBC\_PDE}(\text{pde}, \text{pde.labels}(i), 1, \texttt{'Dirichlet'}, 0., \varnothing)$
end for
$\boldsymbol{X} \leftarrow \text{SolvePDE}(\text{PDE})$

We give in Listings 15 and 16 the corresponding Matlab/Octave and Python codes.

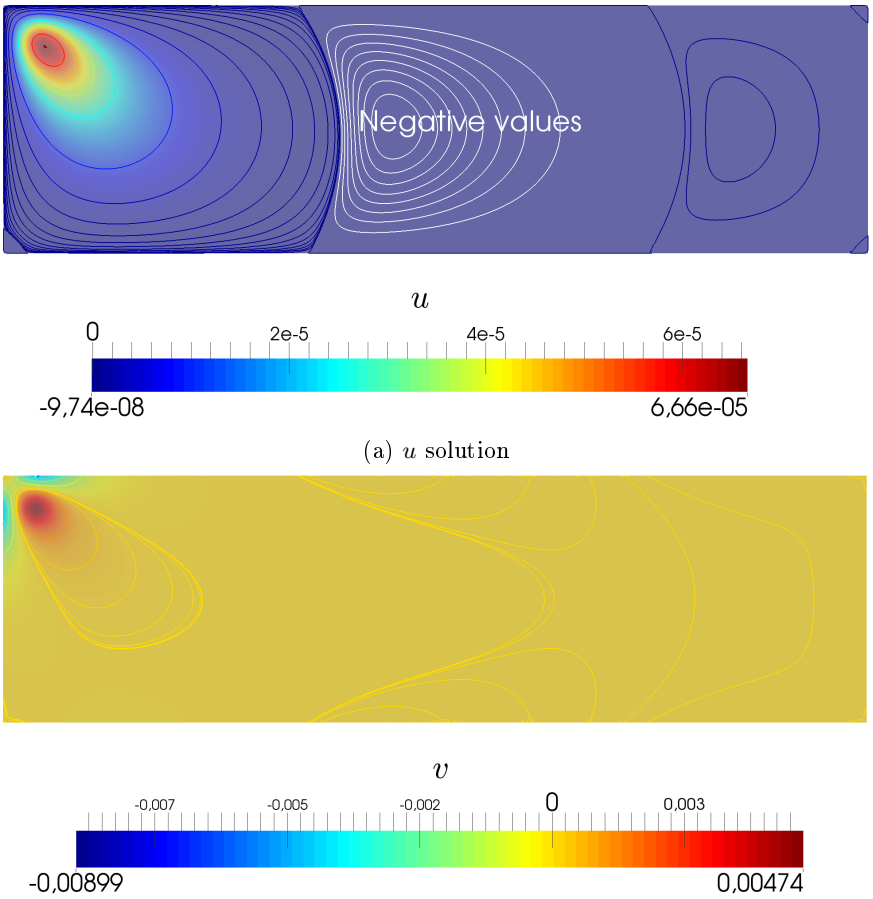
Listing 15: **2D clamped plate, Matlab/Octave code**

```
d=2;m=2;
fprintf('1. Reading of the
    mesh\n');
Th=HyperCube(d,50*[7,2],
    @(q)[7*q(1,:)-1;2*q(2,:)-1]);
fprintf('2. Definition of
    the BVP\n')
Hop=Hoperator(d,m);
Hop.H{1,2}=Loperator(d,{1,0;0,1},[],[],[]);
Hop.H{2,1}=Loperator(d,{1,0;0,1},[],[],[]);
Hop.H{2,2}=Loperator(d,[],[],[],-1);
pde=initPDE(Hop,Th);
pde.f={@(x,y)exp(-100*((x+0.75).^2
    +(y-0.75).^2)),0};
for i=1:pde.nlab
  pde=setBC_PDE(pde,pde.labels(i),1,
    'Dirichlet',0);
end
fprintf('3. Solving BVP\n')
W=solvePDE(pde,'split',true);
```

Listing 16: **2D clamped plate, Python code**

```
d=2;m=2;
print('1. Reading of the
    mesh');
Th=HyperCube(d,[20*7,20*2],trans=lambda
    q:np.c_[7*q[:,0]-1,2*q[:,1]-1])
print('2. Definition of the
    BVP')
Hop=Hoperator(d=2,m=2)
Hop.H[0][1]=Loperator(d=2,
    A=[[1,None],[None,1]])
Hop.H[1][0]=Loperator(d=2,
    A=[[1,None],[None,1]])
Hop.H[1][1]=Loperator(d=2,a0=-1)
pde=initPDE(Hop,Th)
pde.f=[lambda
    x,y:exp(-100*((x+0.75)**2
    +(y-0.75)**2)),0]
for l in pde.labels:
  pde=setBC_PDE(pde,l,0,'Dirichlet',0,None)
print('3. Solving BVP')
x=solvePDE(pde,split=True)
```

The numerical solution for a given mesh is shown on Figure 18



(a) u solution

(b) v solution

Figure 18: Clamped plate problem

2.5 Eigenvalues

3 Eigenvalue problems

We want to solve eigenvalue problems coming from scalar or vector BVP's.

3.1 Scalar case

The eigenvalue problems associated with *scalar* BVP (1.2)-(1.4) can be written as

***Scalar* eigenvalue problem**

Find $(\lambda, u) \in \mathbb{K} \times \mathrm{H}^2(\Omega)$ such that

$$\mathcal{L}(u) = \lambda \mathcal{B}(u) \quad \text{in } \Omega, \tag{3.1}$$

$$u = 0 \quad \text{on } \Gamma^D, \tag{3.2}$$

$$\frac{\partial u}{\partial n_{\mathcal{L}}} + a^R u = 0 \quad \text{on } \Gamma^R. \tag{3.3}$$

where $\mathcal{B} = \mathcal{L}_{\mathbb{O}_{d\times d}, \mathbf{0}_d, \tilde{\mathbf{c}}, \tilde{a_0}}$.

We briefly describe the main function that will be used to solve eigenvalues problems.

$(\boldsymbol{U}, \boldsymbol{\lambda}) \leftarrow$ **EigsPDE**(pde, $\mathcal{B}$, N_e) : compute the N_e smallest magnitude eigenvalues and the corresponding eigenvectors. $\boldsymbol{\lambda}(i)$ is the i-th eigenvalue and its eigenvector is given by $\boldsymbol{U}(:, i)$. The argument pde is obtain by using functions initPDE and setBC_PDE

3.1.1 2D Laplace eigenvalues problem with Dirichlet boundary condition

Let $\Omega \subset \mathbb{R}^2$ be the unit disk given in Figure 1.

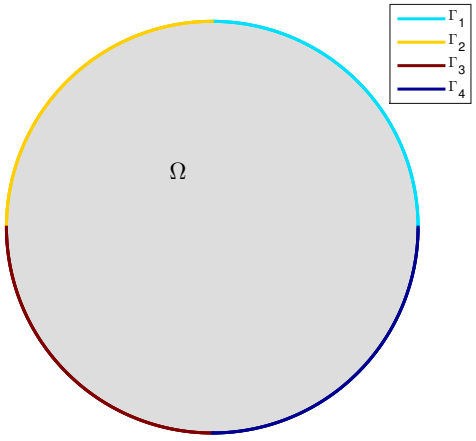


Figure 1: Unit disk with four boundaries

We want to solve the eigenvalue problem given by (3.4)-(3.5)).

2D Laplace eigenvalues problem with Dirichlet boundary condition

Find $(\lambda, u) \in \mathbb{K} \times \mathrm{H}^2(\Omega)$ such that

$$-\Delta u = \lambda u \quad \text{in } \Omega, \tag{3.4}$$
$$u = 0 \quad \text{on } \Gamma, \tag{3.5}$$

This problem is equivalent to the *scalar* eigenvalue problem (3.1)-(3.3) if we set $\mathcal{L} = \mathcal{L}_{\mathbb{I},\mathbf{0},\mathbf{0},0}$, $\mathcal{B} = \mathcal{L}_{\mathbb{O},\mathbf{0},\mathbf{0},1}$, $\Gamma^D = \partial\Omega$ and $\Gamma^R = \varnothing$. The algorithm using the toolbox to find eigenvalues and eigenvectors of this problem is the following :

Algorithm 3.1 2D Laplace eigenvalues problem with Dirichlet boundary condition

1: $\mathcal{T}_h \leftarrow$ getMesh(...) ▷ Load FreeFEM++ mesh
2: $\mathcal{L} \leftarrow$ Loperator$(\mathbb{I}, \mathbf{0}, \mathbf{0}, 0)$ ▷ Laplacian operator
3: pde $\leftarrow$ initPDE$(\mathcal{L}, \mathcal{T}_h)$
4: **for** $i \leftarrow 1$ **to** 4 **do** ▷ Set Dirichlet boundaries conditions
5: pde $\leftarrow$ setBCLabel(pde, `'Dirichlet'`, $i, 1, 0.$) ▷ $u = 0$ on Γ_i
6: **end for**
7: $\mathcal{B} \leftarrow$ Loperator$(\mathbb{O}, \mathbf{0}, \mathbf{0}, 1)$ ▷ set $\mathcal{B}$ operator
8: $(\boldsymbol{U}, \boldsymbol{\lambda}) \leftarrow$ EigsPDE(pde, $\mathcal{B}, 24$)

We represent in Figure 2 eigenvectors associated to the first twenty-four smallest magnitude eigenvalues.

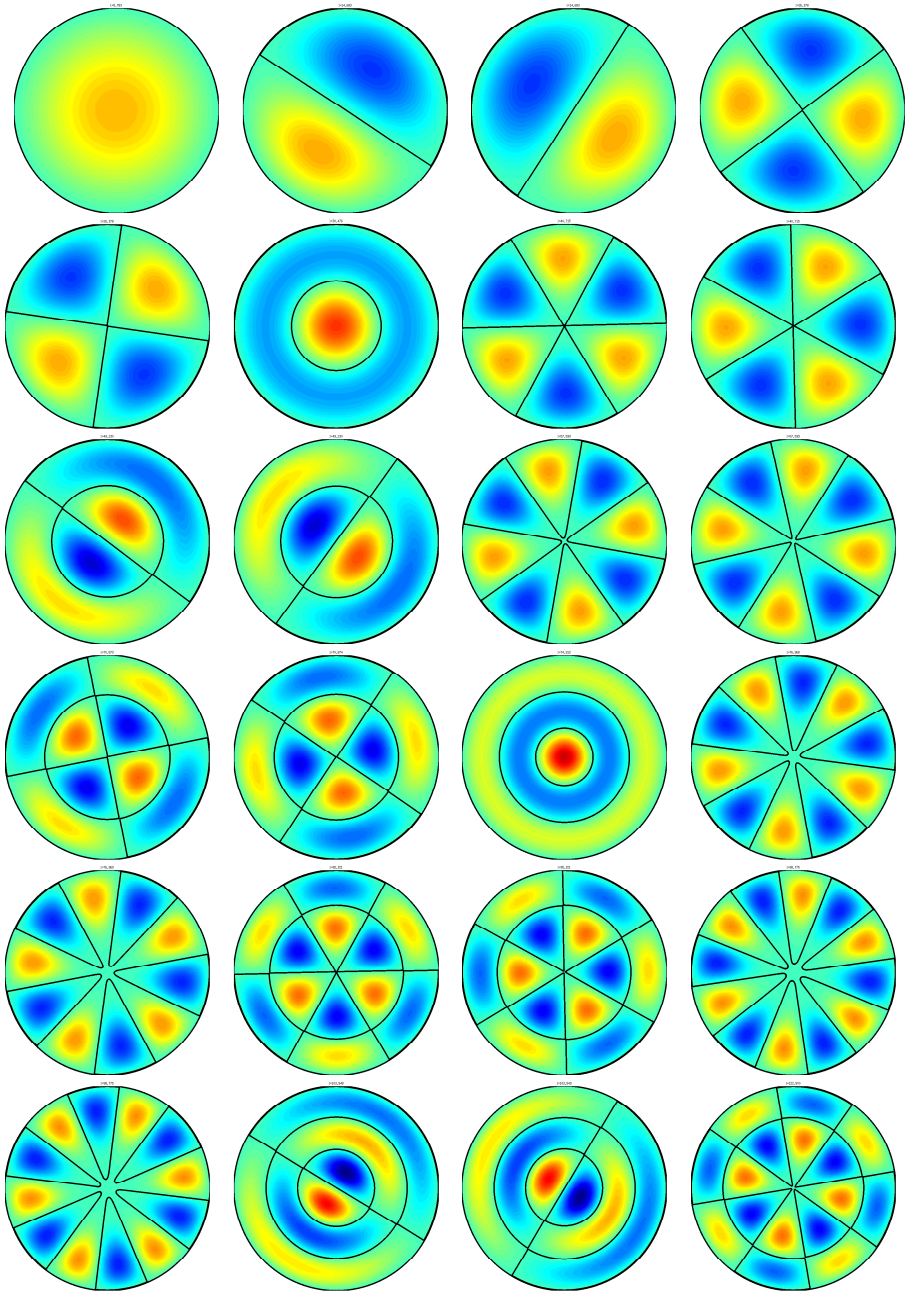


Figure 2: 2D Laplace with Dirichlet boundary conditions : eigenvectors of the smallest magnitude eigenvalues

3.1.2 2D Laplace eigenvalues problem with mixed boundary conditions

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 12).

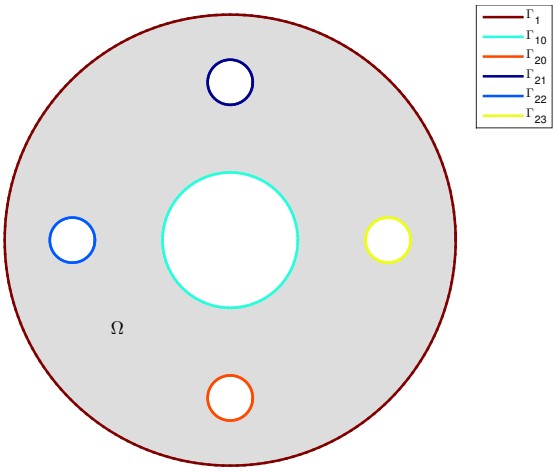


Figure 3: Domain and boundaries

We want to solve the eigenvalue problem given by (3.4)-(??)).

2D Laplace eigenvalues problem with mixed boundary conditions

Find $(\lambda, u) \in \mathbb{K} \times \mathrm{H}^2(\Omega)$ such that

$$-\Delta u = \lambda u \qquad \text{in } \Omega, \tag{3.6}$$

$$u = 0 \qquad \text{on } \Gamma_i, \ \forall i \in [\![20, 23]\!], \tag{3.7}$$

$$\frac{\partial u}{\partial n} = 0 \qquad \text{on } \Gamma_1 \cup \Gamma_{10}. \tag{3.8}$$

This problem is equivalent to the *scalar* eigenvalue problem (3.1)-(3.3) if we set $\mathcal{L} = \mathcal{L}_{\mathbb{I},\mathbf{0},\mathbf{0},0}$, $\mathcal{B} = \mathcal{L}_{\mathbb{O},\mathbf{0},\mathbf{0},1}$, $\Gamma^D = \partial\Omega$ and $\Gamma^R = \varnothing$. The algorithm using the toolbox to find the twenty four smallest magnitude eigenvalues and the corresponding eigenvectors of this problem is the following :

Algorithm 3.2 2D Laplace eigenvalues problem with mixed boundary conditions

1: $\mathcal{T}_h \leftarrow$ getMesh(...) ▷ Load FreeFEM++ mesh
2: $\mathcal{L} \leftarrow$ Loperator($\mathbb{I}, \mathbf{0}, \mathbf{0}, 0$) ▷ Laplacian operator
3: pde $\leftarrow$ initPDE($\mathcal{L}, \mathcal{T}_h$)
4: **for** $i \leftarrow 20$ **to** 23 **do** ▷ Set Dirichlet boundaries conditions
5: pde $\leftarrow$ setBCLabel(pde, `'Dirichlet'`, $i, 1, 0.$) ▷ $u = 0$ on Γ_i
6: **end for**
7: $\mathcal{B} \leftarrow$ Loperator($\mathbb{O}, \mathbf{0}, \mathbf{0}, 1$) ▷ set $\mathcal{B}$ operator
8: $(\boldsymbol{U}, \boldsymbol{\lambda}) \leftarrow$ EigsPDE(pde, $\mathcal{B}, 24$)

We represent in Figure 4 these twenty four eigenvectors.

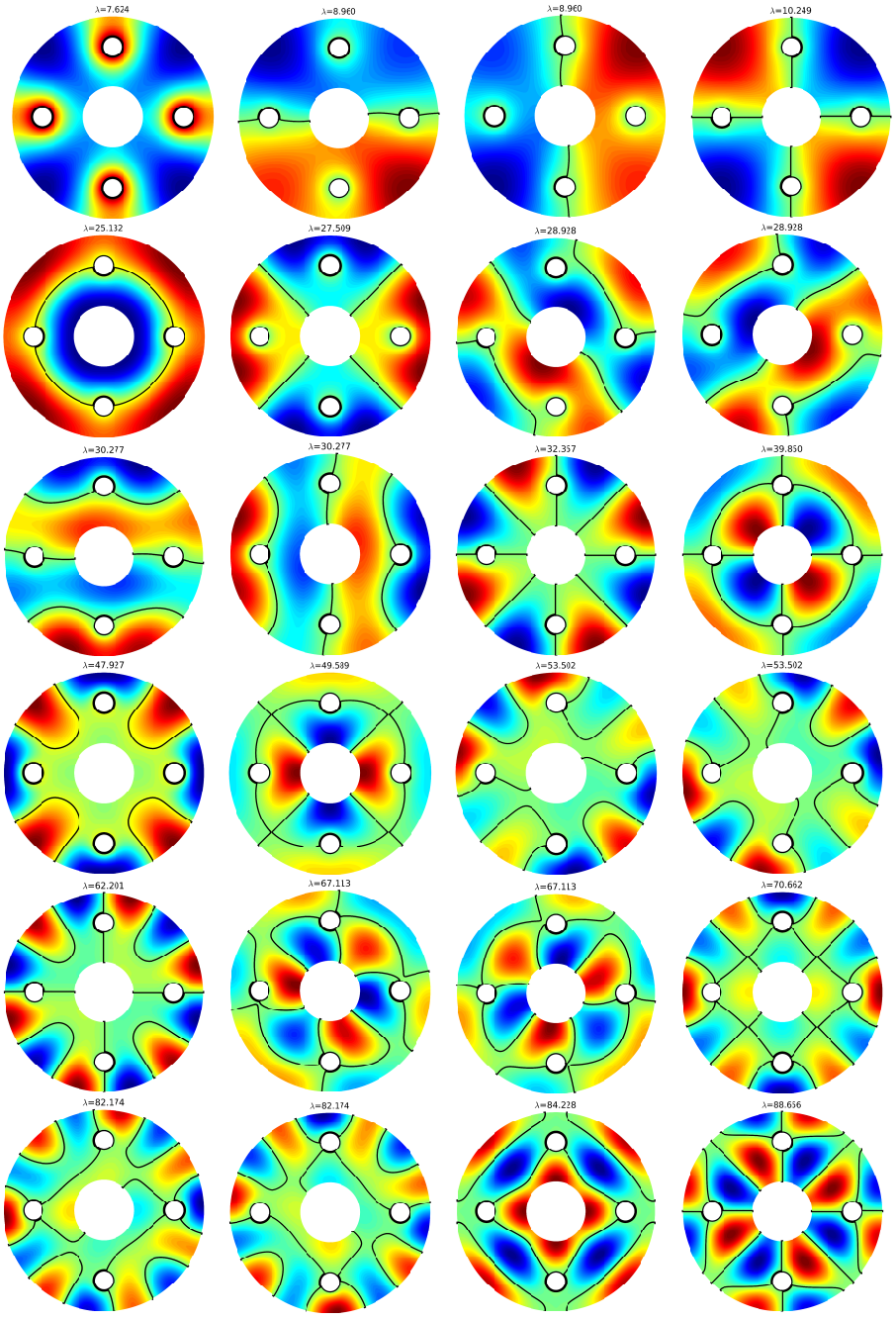


Figure 4: 2D Laplace with mixed boundary conditions : eigenvectors of the smallest magnitude eigenvalues

3.2 Vector case

The eigenvalue problems associated with *vector* BVP (1.10)-(1.12) can be written as

Vector eigenvalue problems

Find $\lambda \in \mathbb{K}$ and $\boldsymbol{u} = (\boldsymbol{u}_1, \ldots, \boldsymbol{u}_m) \in (\mathrm{H}^2(\Omega))^m$ such that

$$\mathcal{H}(\boldsymbol{u}) = \lambda \mathcal{B}(\boldsymbol{u}) \quad \text{in } \Omega, \tag{3.9}$$

$$\boldsymbol{u}_\alpha = 0 \quad \text{on } \Gamma_\alpha^D, \ \forall \alpha \in [\![1, m]\!], \tag{3.10}$$

$$\frac{\partial \boldsymbol{u}}{\partial n_{\mathcal{H}_\alpha}} + a_\alpha^R \boldsymbol{u}_\alpha = 0 \quad \text{on } \Gamma_\alpha^R, \ \forall \alpha \in [\![1, m]\!], \tag{3.11}$$

where $\mathcal{B}$ is a given $\mathcal{H}$-operator.

In most cases $\mathcal{B}$ is the identity operator ($\mathcal{B}$ is a diagonal $\mathcal{H}$-operator with $\mathcal{B}_{\alpha,\alpha} = \mathcal{L}_{\mathbb{O}_{d\times d}, \mathbf{0}_d, \mathbf{0}_d, 1}$, $\forall \alpha \in [\![1, m]\!]$).

References

[1] G. Dhatt, E. Lefrançois, and G. Touzot. *Finite Element Method*. Wiley, 2012.

[2] T. Gerasimov. The clamped elastic grid, a fourth order equation on a domain with corner, 2014.

[3] F. Hecht. New development in freefem++. *J. Numer. Math.*, 20 (3-4):251–265, 2012.

[4] F. Hecht. Freefem++. `www.freefem.org/ff++/index.htm`, 2014.

[5] A. Quarteroni and A. Valli. *Numerical Approximation of Partial Differential Equations*. Springer, 2008.