



fc vfemP1 Matlab toolbox, User's Guide¹
version 0.2.0

François Cuvelier²

Friday 6th March, 2020

¹L^AT_EX manual, revision 0.2.0, compiled with Matlab 2019a, and toolboxes `fc-vfemp1`[0.2.0], `fc-tools`[0.0.31], `fc-bench`[0.1.2], `fc-hypermesh`[1.0.3], `fc-amat`[0.1.1], `fc-meshtools`[0.1.3], `fc-graphics4mesh`[0.1.2], `fc-oogmsh`[0.2.3], `fc-siplt`[0.2.1], `fc-simesh`[0.4.1]

²Université Sorbonne Paris Nord, LAGA, CNRS, UMR 7539, F-93430, Villetaneuse, France, cuvelier@math.univ-paris13.fr.

This work was supported by the ANR project DEDALES under grant ANR-14-CE23-0005.

`vfemP1` is an object-oriented Matlab toolbox dedicated to solve scalar or vector boundary value problem (BVP) by \mathbb{P}^1 -Lagrange finite element methods in any space dimension. It uses the `FC-SIMESH` toolbox and more particularly the `siMeshobject` which allows to use simplices meshes generated from `gmsh` (in dimension 2 or 3) or an hypercube triangulation (in any dimension). For graphical representation (dimension ≤ 3 !) the `FC-SIPLT` toolbox is used.

This toolbox also contains the techniques of vectorization presented in [4] and extended in [3] and allows good performances when using finite elements methods.

Contents

1	Presentation	5
1.1	Installation	5
1.1.1	Installation automatic, all in one (recommanded)	5
1.1.2	Other kind	7
1.2	Scalar boundary value problem	7
1.2.1	Solving scalar BVP: condenser problem	8
1.3	Vector boundary value problem	9
1.3.1	Solving vector BVP: funny problem	10
2	Objects	13
2.1	fc_vfemp1.Fdata object	13
2.2	fc_vfemp1.Loperator object	13
2.2.1	Constructor	14
2.2.2	apply method	15
2.3	fc_vfemp1.Hoperator object	16
2.3.1	Constructor	17
2.3.2	apply method	18
2.3.3	set method	18
2.4	fc_vfemp1.PDE object	18
2.5	fc_vfemp1.BVP object	18
2.5.1	Constructor	19
2.5.2	setPDE method	19
2.5.3	Description	19
2.5.4	setDirichlet method	19
2.5.5	setRobin method	20
2.5.6	solve method	20
2.5.7	Assembly method	20
3	Scalar boundary value problems	23
3.1	Poisson BVP's	23
3.1.1	2D Poisson BVP with Dirichlet boundary conditions on the unit square	23
3.1.2	2D Poisson BVP with mixed boundary conditions	27
3.1.3	3D Poisson BVP with mixed boundary conditions	30
3.1.4	4D Poisson BVP with mixed boundary conditions	34

3.1.5	1D BVP : just for fun	37
3.2	Stationary convection-diffusion problem	37
3.2.1	Stationary convection-diffusion problem in 2D	37
3.2.2	Stationary convection-diffusion problem in 3D	40
3.3	2D electrostatic BVPs	42
4	Vector boundary value problems	47
4.1	Elasticity problem	47
4.1.1	General case ($d = 2, 3$)	47
4.1.2	2D example	49
4.1.3	3D example	50
4.2	Stationary heat with potential flow in 2D	52
4.2.1	Method 1 : split in three parts	55
4.2.2	Method 2 : have fun with \mathcal{H} -operators	56
4.3	Stationary heat with potential flow in 3D	57
4.3.1	Method 1 : split in three parts	60
4.3.2	Method 2 : have fun with \mathcal{H} -operators	61
5	Other problems	65
	Appendices	69
1	Linear elasticity	69
1.1	Elasticity in \mathbb{R}^d	69

Chapter 1

Presentation

Firstly, the installation process of the vcfem^{P1} toolbox is presented. Thereafter a generic scalar boundary value problem is given by using notations of [6]. A simple example of such problem is solved with this toolbox. A last, by extended previous notations, a generic vector boundary value problem is described and an example solved.

1.1 Installation

1.1.1 Installation automatic, all in one (recommended)

For this method, one just have to get/download the install file

```
mfc_vfemp1_install.m
```

or get it on the dedicated web page. Thereafter, one run it under Matlab. This command download, extract and configure the *fc-vfemp1* and the required toolbox *fc-simesh* in the current directory.

For example, to install this toolbox in `~/Matlab/toolboxes` directory, one have to copy the file `mfc_vfemp1_install.m` in the `~/Matlab/toolboxes` directory. Then in a Matlab terminal run the following commands

```
>> cd ~/Matlab/toolboxes
>> mfc_vfemp1_install()
```

There is the output of the `mfc_vfemp1_install()` command on a Linux computer:

```

Parts of the Matlab <fc-vfemp1> toolbox.
Copyright (C) 2016-2017 F. Cuvelier <cuvelier@math.univ-paris13.fr>

*****
Downloading and installing the toolbox
<fc-simesh>[0.2.1]
*****
Parts of the Matlab <fc-simesh> toolbox.
Copyright (C) 2016-2017 F. Cuvelier <cuvelier@math.univ-paris13.fr>

1- Downloading and extracting the toolboxes
-> <fc-tools>[0.0.19] ... OK
-> <fc-hypermesh>[0.0.6] ... OK
-> <fc-oogmsh>[0.0.17] ... OK
-> <fc-simesh>[0.2.1] ... OK
-> <fc-graphics4mesh>[0.0.2] ... OK
-> <fc-siplt>[0.0.2] ... OK
2- Setting the toolboxes
2-a) Setting the <fc-hypermesh> toolbox
Write in ~/Matlab/toolboxes/fc-vfemp1-full/fc_hypermesh-0.0.6/configure_loc.m ...
-> done
2-b) Setting the <fc-oogmsh> toolbox
[fc-oogmsh] Using GMSH binary : ~/bin/gmsh
[fc-oogmsh] Writing in ~/Matlab/toolboxes/fc-vfemp1-full/fc_oogmsh-0.0.17/configure_loc.m ...
[fc-oogmsh] configured with
-> gmsh_bin='~/bin/gmsh';
-> mesh_dir='~/Matlab/toolboxes/fc-vfemp1-full/fc_oogmsh-0.0.17/meshes';
-> geo_dir='~/Matlab/toolboxes/fc-vfemp1-full/fc_oogmsh-0.0.17/geodir';
-> fc_tools_dir='~/Matlab/toolboxes/fc-vfemp1-full/fc_tools-0.0.19';
[fc-oogmsh] done
2-c) Setting the <fc-simesh> toolbox without graphics
2-d) Setting the <fc-graphics4mesh> toolbox
Write in ~/Matlab/toolboxes/fc-vfemp1-full/fc_graphics4mesh-0.0.2/configure_loc.m ...
-> done
2-e) Setting the <fc-siplt> toolbox
Write in ~/Matlab/toolboxes/fc-vfemp1-full/fc_siplt-0.0.2/configure_loc.m ...
-> done
2-f) Setting the <fc-simesh> toolbox with graphics
[fc-simesh] Writing in ~/Matlab/toolboxes/fc-vfemp1-full/fc_simesh-0.2.1/configure_loc.m ...
[fc-simesh] configured with
-> oogmsh_dir = '~/Matlab/toolboxes/fc-vfemp1-full/fc_oogmsh-0.0.17';
-> hypermesh_dir = '~/Matlab/toolboxes/fc-vfemp1-full/fc_hypermesh-0.0.6';
-> siplt_dir = '~/Matlab/toolboxes/fc-vfemp1-full/fc_siplt-0.0.2';
[fc-simesh] done
3- Using instructions
To use the <fc-simesh> toolbox:
addpath('~/Matlab/toolboxes/fc-vfemp1-full/fc_simesh-0.2.1')
fc_simesh.init()

See ~/Matlab/toolboxes/mfc_simesh_set.m
<fc-simesh>[0.2.1]: installed

*****
Downloading and installing the toolbox
<fc-vfemp1>[0.1.0]
*****
*** Setting the <fc-vfemp1> toolbox
Write in ~/Matlab/toolboxes/fc-vfemp1-full/fc_vfemp1-0.1.0/configure_loc.m ...
-> done
*** The <fc-vfemp1>[0.1.0] is installed
*** Using instructions
To use the <fc-vfemp1> toolbox:
addpath('~/Matlab/toolboxes/fc-vfemp1-full/fc_vfemp1-0.1.0')
fc_vfemp1.init()

See ~/Matlab/toolboxes/mfc_vfemp1_set.m

```

The complete toolbox (i.e. with all the other needed toolboxes) is stored in the directory

~/Matlab/toolboxes/fc-vfemp1-full

and, for each Matlab session, one have to set the toolbox by:

```
>> addpath('~/Matlab/toolboxes/fc-vfemp1-full/fc-vfemp1-0.1.0')
>> fc_vfemp1.init()
```

To quickly test this toolbox, one can run one of the script examples located in the directory +fc_vfemp1/+examples. For example, runs

```
>> fc_vfemp1.examples.BVPCondenser2D01
```

or the complete demo (take a long time)

```
>> fc_vfemp1.demos()
```

To install the *fc-vfemp1* toolbox without graphical extension one can use the following command

```
>> mfc_vfemp1_install('graphics',false)
```

For **uninstalling**, one just have to delete directory

```
~/Matlab/toolboxes/fc-vfemp1-full
```

1.1.2 Other kind

to do!

1.2 Scalar boundary value problem

The notations of [6] are employed in this section and extended to the vector case. Let Ω be a bounded open subset of \mathbb{R}^d , $d \geq 1$. The boundary of Ω is denoted by Γ .

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

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

where $\mathbb{A} \in (L^\infty(\Omega))^{d \times d}$, $\mathbf{b} \in (L^\infty(\Omega))^d$, $\mathbf{c} \in (L^\infty(\Omega))^d$ 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 [6] and we note that we can omit either $\operatorname{div}(\mathbf{b}u)$ or $\langle \nabla u, \mathbf{c} \rangle$ if \mathbf{b} and \mathbf{c} are sufficiently regular functions. It should be also noted that it is important to preserve the two terms \mathbf{b} and \mathbf{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 $f \in L^2(\Omega)$, $g^D \in \mathbb{H}^{1/2}(\Gamma^D)$, $g^R \in L^2(\Gamma^R)$, $a^R \in L^\infty(\Gamma^R)$ be given data.

A *scalar* boundary value problem is given by

Scalar BVP 1 : generic problem

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

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

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

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

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

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

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

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

$$\mathcal{D}_{\mathcal{L}}(u, v) = \langle \mathbb{A} \nabla u, \nabla v \rangle - (u \langle \mathbf{b}, \nabla v \rangle - v \langle \nabla u, \mathbf{c} \rangle) + a_0 uv. \quad (1.6)$$

A variational formulation associated to the scalar boundary value problem (1.2)-(1.4) reads

🔗 Scalar VF 1 : generic problem

Find $u \in H_{g^D, \Gamma^D}^1(\Omega)$ such that

$$\mathcal{A}_{\mathcal{L}}(u, v) = \mathcal{F}(v), \quad \forall v \in H_{0, \Gamma^D}^1(\Omega) \quad (1.7)$$

where

$$\mathcal{A}_{\mathcal{L}}(u, v) = \int_{\Omega} \mathcal{D}_{\mathcal{L}}(u, v) d\mathbf{q} + \int_{\Gamma^R} a^R u v d\sigma \quad (1.8)$$

$$\mathcal{F}(v) = \int_{\Omega} f v d\mathbf{q} + \int_{\Gamma^R} g^R v d\sigma \quad (1.9)$$

To have an outline of the [CvFEM_{P1}](#) toolbox, a first and simple problem is quickly present. Explanations will be given in next chapters.

1.2.1 Solving scalar BVP: condenser problem

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

💡 Usual BVP 1 : 2D condenser problem

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

$$-\Delta u = 0 \quad \text{in } \Omega \subset \mathbb{R}^2, \quad (1.10)$$

$$u = 0 \quad \text{on } \Gamma_1, \quad (1.11)$$

$$u = -12 \quad \text{on } \Gamma_{98}, \quad (1.12)$$

$$u = 12 \quad \text{on } \Gamma_{99}, \quad (1.13)$$

where Ω and its boundaries are given in Figure 1.1.

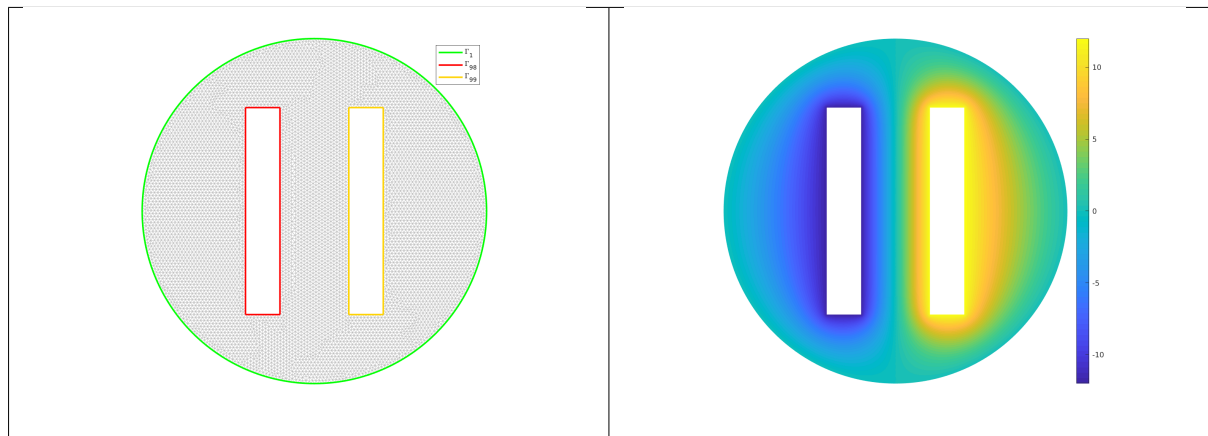


Figure 1.1: 2D condenser mesh and boundaries (left) and numerical solution (right)

The problem (1.10)-(1.13) can be equivalently expressed as the scalar BVP (1.2)-(1.4) :

🔗 Scalar BVP 2 : 2D condenser problem

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

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

where $\mathcal{L} := \mathcal{L}_{\mathbb{1}, \mathbf{0}, \mathbf{0}, 0}$, $f \equiv 0$, and

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

In Listing 1.1 a complete code is given to solve this problem and in Table 1.1 computational times for assembling and solving steps are given with various size meshes.

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

```
meshfile=fc_oogmsh.buildmesh2d('condenser',10); % generate mesh
Th=fc_simesh.siMesh(meshfile); % read mesh
Lop=fc_vfemp1.Loperator(2,2,{1,0;0,1},[],[],[]);
pde=fc_vfemp1.PDE(Lop);
bvp=fc_vfemp1.BVP(Th,pde);
bvp.setDirichlet(1,0);
bvp.setDirichlet(98,-12.);
bvp.setDirichlet(99,+12.);
U=bvp.solve();
% Graphic parts
figure(1)
Th.plotmesh('color',0.7*[1,1,1])
hold on;axis off,axis image;
Th.plotmesh('d',1,'Linewidth',2,'inlegend',true)
legend('show')
figure(2)
Th.plot(U,'edgecolor','none','facecolor','interp')
axis off,axis image;colorbar
```

N	n_q	n_{me}	Assembly	Solve
100	10 201	20 000	0.124 (s)	0.029 (s)
200	40 401	80 000	0.179 (s)	0.130 (s)
300	90 601	180 000	0.404 (s)	0.383 (s)
400	160 801	320 000	0.741 (s)	0.699 (s)
500	251 001	500 000	1.164 (s)	1.150 (s)
600	361 201	720 000	1.711 (s)	1.796 (s)
700	491 401	980 000	2.314 (s)	2.847 (s)
800	641 601	1 280 000	3.255 (s)	4.034 (s)

Table 1.1: Computational times for assembling and solving the 2D condenser BVP, described in *Scalar BVP 2*, with various size meshes.

Obviously, more complex problems will be studied in chapter 3 and complete explanations on the code will be given in next chapters. Previously, the vector BVP is formally presented with an application.

1.3 Vector boundary value problem

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

$$\begin{cases} \mathcal{H} : (\mathbb{H}^2(\Omega))^m & \longrightarrow (L^2(\Omega))^m \\ \mathbf{u} = (\mathbf{u}_1, \dots, \mathbf{u}_m) & \longmapsto \mathbf{f} = (\mathbf{f}_1, \dots, \mathbf{f}_m) \stackrel{\text{def}}{=} \mathcal{H}(\mathbf{u}) \end{cases} \quad (1.14)$$

where

$$\mathbf{f}_\alpha = \sum_{\beta=1}^m \mathcal{H}_{\alpha,\beta}(\mathbf{u}_\beta), \quad \forall \alpha \in \llbracket 1, m \rrbracket, \quad (1.15)$$

with, for all $(\alpha, \beta) \in \llbracket 1, m \rrbracket^2$,

$$\mathcal{H}_{\alpha,\beta} \stackrel{\text{def}}{=} \mathcal{L}_{\mathbb{A}^{\alpha,\beta}, \mathbf{b}^{\alpha,\beta}, \mathbf{c}^{\alpha,\beta}, a_0^{\alpha,\beta}} \quad (1.16)$$

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

$$\mathcal{H}(\mathbf{u}) = \begin{pmatrix} \mathcal{L}_{\mathbb{A}^{1,1}, \mathbf{b}^{1,1}, \mathbf{c}^{1,1}, a_0^{1,1}} & \cdots & \mathcal{L}_{\mathbb{A}^{1,m}, \mathbf{b}^{1,m}, \mathbf{c}^{1,m}, a_0^{1,m}} \\ \vdots & \ddots & \vdots \\ \mathcal{L}_{\mathbb{A}^{m,1}, \mathbf{b}^{m,1}, \mathbf{c}^{m,1}, a_0^{m,1}} & \cdots & \mathcal{L}_{\mathbb{A}^{m,m}, \mathbf{b}^{m,m}, \mathbf{c}^{m,m}, a_0^{m,m}} \end{pmatrix} \begin{pmatrix} \mathbf{u}_1 \\ \vdots \\ \mathbf{u}_m \end{pmatrix}. \quad (1.17)$$

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

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

A *vector* boundary value problem is given by

Vector BVP 1 : generic problem

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

$$\mathcal{H}(\mathbf{u}) = \mathbf{f} \quad \text{in } \Omega, \quad (1.18)$$

$$\mathbf{u}_\alpha = g_\alpha^D \quad \text{on } \Gamma_\alpha^D, \quad \forall \alpha \in \llbracket 1, m \rrbracket, \quad (1.19)$$

$$\frac{\partial \mathbf{u}}{\partial n_{\mathcal{H}_\alpha}} + a_\alpha^R \mathbf{u}_\alpha = g_\alpha^R \quad \text{on } \Gamma_\alpha^R, \quad \forall \alpha \in \llbracket 1, m \rrbracket, \quad (1.20)$$

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

$$\frac{\partial \mathbf{u}}{\partial n_{\mathcal{H}_\alpha}} \stackrel{\text{def}}{=} \sum_{\beta=1}^m \frac{\partial \mathbf{u}_\beta}{\partial n_{\mathcal{H}_{\alpha,\beta}}} = \sum_{\beta=1}^m \left(\langle \mathbb{A}^{\alpha,\beta} \nabla \mathbf{u}_\beta, \mathbf{n} \rangle - \langle \mathbf{b}^{\alpha,\beta} \mathbf{u}_\beta, \mathbf{n} \rangle \right). \quad (1.21)$$

The boundary conditions (1.20) are the **Robin** boundary conditions and (1.19) 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 $\frac{\partial \mathbf{u}}{\partial n_{\mathcal{H}_1}} + a_1^R \mathbf{u}_1 = g_1^R$ and a Dirichlet one with $\mathbf{u}_2 = g_2^D$.

A variational form of the *vector* BVP (1.18)-(1.20) is given by

Variational form of the *vector* BVP

Find $\mathbf{u} \in H_{g_1^D, \Gamma_1^D}^1 \times \dots \times H_{g_m^D, \Gamma_m^D}^1$ such that

$$\mathbf{A}_{\mathcal{H}}(\mathbf{u}, \mathbf{v}) = \mathcal{F}(\mathbf{v}) \quad \forall \mathbf{v} \in H_{0, \Gamma_1^D}^1 \times \dots \times H_{0, \Gamma_m^D}^1 \quad (1.22)$$


where

$$\mathbf{A}_{\mathcal{H}}(\mathbf{u}, \mathbf{v}) = \int_{\Omega} \mathcal{D}_{\mathcal{H}}(\mathbf{u}, \mathbf{v}) d\mathbf{q} + \sum_{\alpha=1}^m \int_{\Gamma_\alpha^R} a_\alpha^R \mathbf{u}_\alpha \mathbf{v}_\alpha d\sigma \quad (1.23)$$

$$\mathcal{F}(\mathbf{v}) = \int_{\Omega} \langle \mathbf{f}, \mathbf{v} \rangle d\mathbf{q} + \sum_{\alpha=1}^m \int_{\Gamma_\alpha^R} g_\alpha^R \mathbf{v}_\alpha d\sigma \quad (1.24)$$

where

$$\mathcal{D}_{\mathcal{H}}(\mathbf{u}, \mathbf{v}) = \sum_{\alpha=1}^m \sum_{\beta=1}^m \mathcal{D}_{\mathcal{H}_{\alpha,\beta}}(\mathbf{u}_\beta, \mathbf{v}_\alpha) \quad (1.25)$$

To have an outline of the  toolbox, a second and simple problem is quickly present.

1.3.1 Solving vector BVP: funny problem

Usual vector BVP 1 : 2D simple vector problem

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

$$-\Delta u_1 + u_2 = 0 \text{ in } \Omega \subset \mathbb{R}^2, \quad (1.26)$$

$$-\Delta u_2 + u_1 = 0 \text{ in } \Omega \subset \mathbb{R}^2, \quad (1.27)$$

$$(u_1, u_2) = (0, 0) \text{ on } \Gamma_1, \quad (1.28)$$

$$(u_1, u_2) = (-12., +12.) \text{ on } \Gamma_{98}, \quad (1.29)$$

$$(u_1, u_2) = (+12., -12.) \text{ on } \Gamma_{99}, \quad (1.30)$$

where Ω and its boundaries are given in Figure 1.1.

The problem (1.26)-(1.30) can be equivalently expressed as the vector BVP (1.2)-(1.4) :

Vector BVP 2 : 2D simple vector problem

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

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

where

$$\mathcal{H} := \begin{pmatrix} \mathcal{L}_{1,0,0,0} & \mathcal{L}_{0,0,0,1} \\ \mathcal{L}_{0,0,0,1} & \mathcal{L}_{1,0,0,0} \end{pmatrix}, \text{ as } \mathcal{H} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \begin{pmatrix} -\Delta & 1 \\ 1 & -\Delta \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}$$

$$f \equiv 0,$$

and

$$\begin{aligned} g_1^D &= g_2^D := 0 \text{ on } \Gamma_1, \\ g_1^D &:= -12, \quad g_2^D := +12 \text{ on } \Gamma_{98}, \\ g_1^D &:= +12, \quad g_2^D := -12 \text{ on } \Gamma_{99}. \end{aligned}$$

In Listing 1.2 a complete code is given to solve this problem. Numerical solutions are represented in Figure 1.2. In Table 1.2 computational times for assembling and solving steps are given with various size meshes.

Listing 1.2: Complete Matlab code to solve the simple 2D vector BVP with graphical representations

```
meshfile=fc_oogmsh.buildmesh2d('condenser',10); % generate mesh
Th=fc_simesh.siMesh(meshfile); % read mesh
Hop=fc_vfemp1.Hoperator(2,2,2);
Hop.set([1,2],[1,2],fc_vfemp1.Loperator(2,2,{1,[],[];[] ,1},[],[],[]));
Hop.set([1,2],[2,1],fc_vfemp1.Loperator(2,2,[],[],[],1));
pde=fc_vfemp1.PDE(Hop);
bvp=fc_vfemp1.BVP(Th,pde);
bvp.setDirichlet(1,0,1:2);
bvp.setDirichlet(98,{12,+12},1:2);
bvp.setDirichlet(99,{+12,12},1:2);
U=bvp.solve('split',true);
% Graphic parts
figure(1)
Th.plot(U{1})
axis image;axis off;shading interp
colorbar
figure(2);
Th.plot(U{2})
axis image;axis off;shading interp
colorbar
```

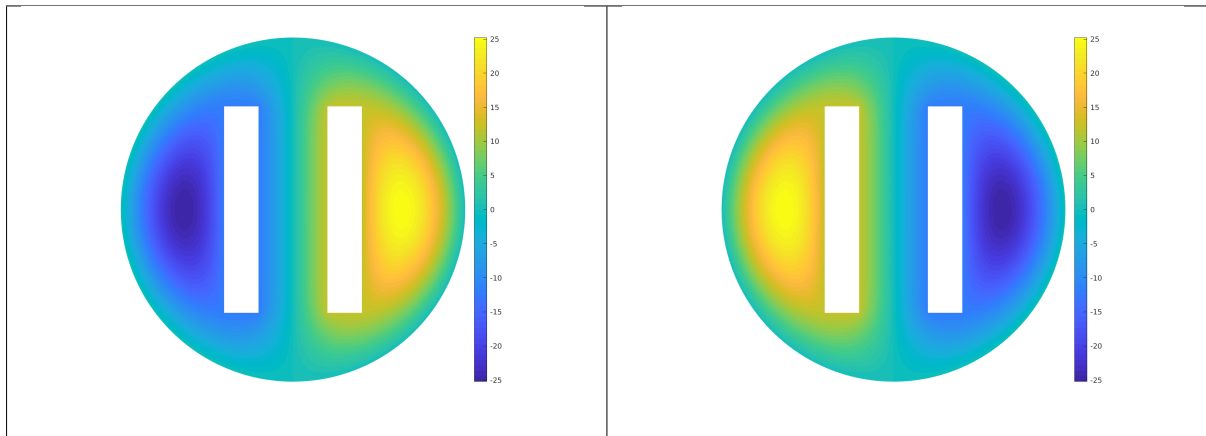


Figure 1.2: 2D simple vector BVP, u_1 numerical solution (left) and u_2 numerical solution (right)

N	n_q	n_{me}	n_{dof}	Assembly	Solve
10	8 151	15 708	16 302	0.202 (s)	0.100 (s)
20	31 742	62 296	63 484	0.417 (s)	0.535 (s)
30	70 744	139 706	141 488	1.081 (s)	1.369 (s)
40	124 930	247 484	249 860	2.101 (s)	2.474 (s)
50	194 775	386 580	389 550	3.582 (s)	4.690 (s)
60	279 962	556 360	559 924	5.499 (s)	7.264 (s)

Table 1.2: Computational times for assembling and solving the 2D simple vector BVP, described in *Scalar BVP 2*, with various size meshes.

Obviously, more complex problems will be studied in chapter 4 and complete explanations on the code will be given in next sections.

In the following of the report we will solve by a \mathbb{P}^1 -Lagrange finite element method *scalar* B.V.P. (1.2) to (1.4) and *vector* B.V.P. (1.18) to (1.20) without additional restrictive assumption.

Chapter 2

Objects

2.1 `fc_vfemp1.Fdata` object

This object is used to create the datas associated with the scalar boundary value problem (1.2)-(1.4) or vector boundary value problem (1.18)-(1.20).

2.2 `fc_vfemp1.Loperator` object

The `fc_vfemp1.Loperator` object is used to create the operator $\mathcal{L}_{\mathbf{A}, \mathbf{b}, \mathbf{c}, a_0}$ defined in (1.1). Its main properties are

Properties of <code>fc_vfemp1.Loperator</code> object	
<code>dim</code>	: integer, space dimension.
<code>d</code>	: integer the operator acts on <code>d</code> -dimensional surfaces.
<code>A</code>	: array of <code>d</code> -by- <code>d</code> cells. Used to store the <code>A</code> functions such that $A\{i,j\} \leftarrow A_{i,j}$. Each cell contains a <code>Fdata</code> object or is empty for 0 value.
<code>b</code>	: array of <code>d</code> -by-1 cells. Used to store the <code>b</code> functions such that $b\{i\} \leftarrow b_i$. Each cell contains a <code>Fdata</code> object or is empty for 0 value.
<code>c</code>	: array of <code>d</code> -by-1 cells. Used to store the <code>c</code> functions such that $c\{i\} \leftarrow c_i$. Each cell contains a <code>Fdata</code> object or is empty for 0 value.
<code>a0</code>	: a <code>Fdata</code> object or empty for 0 value Used to store the <code>a0</code> function such that $a0 \leftarrow a_0$.
<code>order</code>	: integer order of the operator : 2 if <code>A</code> is not empty, 1 if <code>A</code> is empty and <code>b</code> or <code>c</code> not empty, 0 if <code>A</code> , <code>b</code> and <code>c</code> are empty.

2.2.1 Constructor

Its constructor are

```
obj=fc_vfemp1.Loperator()
obj=fc_vfemp1.Lopertor(dim,d,A,b,c,a0)
```

Description

`obj=fc_vfemp1.Loperator()` create an empty operator.

`obj=fc_vfemp1.Loperator(dim,d,A,b,c,a0)` ...

- `dim` is the space dimension
- Usually `d=dim`.
-

Samples

$$-\Delta u := \mathcal{L}_{1,0,0,0}$$

```
in ℝ   Lop=fc_vfemp1.Loperator(1,1,{1},[],[],[])
in ℝ2 Lop=fc_vfemp1.Loperator(2,2,{1,[],1},[],[],[])
in ℝ3 Lop=fc_vfemp1.Loperator(3,3,{1,[],[],1,[],1},[],[],[])
⋮
```

$$-\Delta u + u := \mathcal{L}_{1,0,0,1}$$

```
in ℝ   Lop=fc_vfemp1.Loperator(1,1,{1},[],[],1)
in ℝ2 Lop=fc_vfemp1.Loperator(2,2,{1,[],1},[],[],1)
in ℝ3 Lop=fc_vfemp1.Loperator(3,3,{1,[],[],1,[],1},[],[],1)
⋮
```

In R^2 , $-\Delta u + (1 + \cos(x + y))u := \mathcal{L}_{\mathbf{1}, \mathbf{0}, \mathbf{0}, (x,y) \mapsto (1 + \cos(x+y))}$

`Lop=fc_vfemp1.Loperator (2,2,{1,[],[],1},[],[], @(x,y) 1+cos(x+y))`

2.2.2 apply method

We consider the first order linear operator \mathcal{L}^1 given by

$$\mathcal{L}^1 \stackrel{\text{def}}{=} \mathcal{L}_{\mathbf{0}, \mathbf{0}, \mathbf{c}, a_0}$$

where $\mathbf{c} \in (L^\infty(\Omega))^d$ and $a_0 \in L^\infty(\Omega)$. For a given function $u \in L^2(\Omega)$, the goal of the `apply` method is to compute an approximation of

$$w = \mathcal{L}^1(u) = \langle \mathbf{c}, \nabla u \rangle + a_0 \cdot u.$$

The variational form of this problem is to find $w \in L^2(\Omega)$ such that

$$\int_{\Omega} w \cdot v \, dq = \int_{\Omega} \langle \mathbf{c}, \nabla u \rangle v \, dq + \int_{\Omega} a_0 u v \, dq, \quad \forall v \in L^2(\Omega).$$

Description

`W=Lop.apply(Th,u)`

`Lop` is a `fc_vfemp1.Loperator` object (order 1), `Th` is a `fc_simesh.siMesh` object and `u` is a function or a vector of dimension `Th.nq` (number of nodes of the mesh). Returns `W` a vector of dimension `Th.nq`

Example: computing $\text{div}(u)$

In dimension 2, we have

$$\mathcal{L}_{\mathbf{0}, \mathbf{0}, (1,1)^t, 0}(u) = \text{div}(u).$$

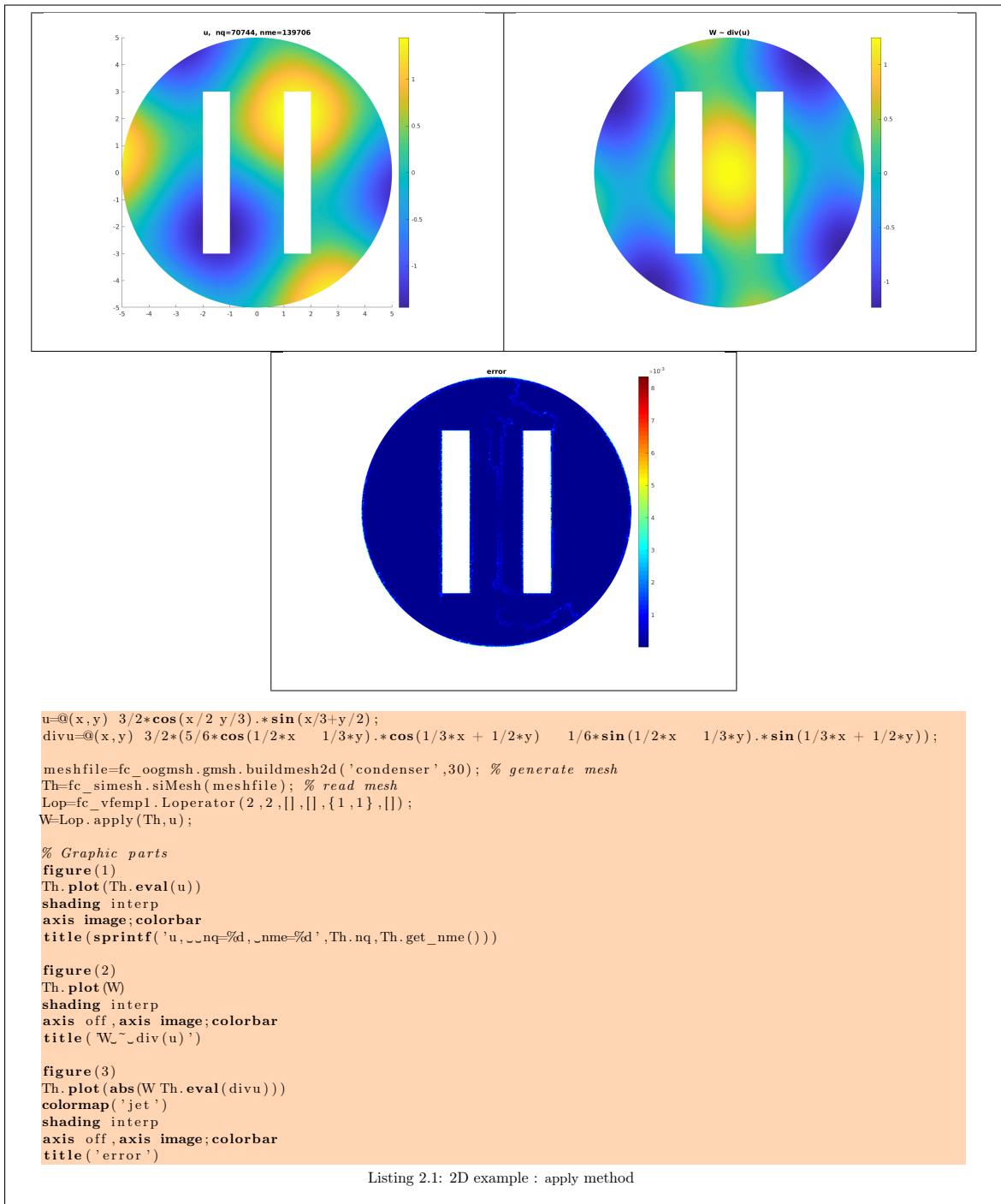
So to create this operator under Matlab one just has to do

`Lop=fc_vfemp1.Loperator (2,2,[],[],{1,1},[]);`

As example, we take $u(x, y) \stackrel{\text{def}}{=} \cos\left(\frac{1}{2}x - \frac{1}{3}y\right) \sin\left(\frac{1}{3}x + \frac{1}{2}y\right)$ and then we have

$$\text{div}(u(x, y)) = \frac{5}{6} \cos\left(\frac{1}{2}x - \frac{1}{3}y\right) \cos\left(\frac{1}{3}x + \frac{1}{2}y\right) - \frac{1}{6} \sin\left(\frac{1}{2}x - \frac{1}{3}y\right) \sin\left(\frac{1}{3}x + \frac{1}{2}y\right)$$

We give in Listing 1 a complete script with graphical representations.



2.3 `fc_vfemp1.Hoperator` object

The object `fc_vfemp1.Hoperator` is used to create a \mathcal{H} operator defined in (1.14). Its main properties are

Properties of `fc_vfemp1.Hoperator` object

<code>dim</code>	: integer, space dimension.
<code>d</code>	: integer the operator acts on <code>d</code> -dimensional surfaces.
<code>m</code>	: integer dimension of the \mathcal{H} operator
<code>H</code>	: array of <code>d</code> -by- <code>d</code> cells. Used to store the \mathcal{H} operators such that $H\{i,j\} \leftarrow \mathcal{H}_{i,j}, \forall i,j \in \llbracket 1,m \rrbracket$. Each cell contains a <code>fc_vfemp1.Loperator</code> object or an empty value.

2.3.1 Constructor

Its constructor are

```
obj=fc_vfemp1.Hoperator()
obj=fc_vfemp1.Hoperator(dim,d,m)
```

Description

`obj=fc_vfemp1.Hoperator()` create an empty operator with all dimensions set to 0.

`obj=fc_vfemp1.Hoperator(dim,d,m)` create an empty/null operator with the given dimensions.

Samples

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

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

could be written as

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

and then

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

```
Hop=fc_vfemp1.Hoperator(2,2,2);
Lop1=fc_vfemp1.Loperator(2,2,{1,[],[]},[],[],[]);
Lop2=fc_vfemp1.Loperator(2,2,[],[],[],1);
Hop.set(1,1,Lop1);Hop.set(2,2,Lop1);
Hop.set(1,2,Lop2);Hop.set(2,1,Lop2);
```

or

```
Hop=fc_vfemp1.Hoperator(2,2,2);
Hop.set([1,2],[1,2],fc_vfemp1.Loperator(2,2,{1,[],[]},[],[],[]));
Hop.set([1,2],[2,1],fc_vfemp1.Loperator(2,2,[],[],[],1));
```

2.3.2 apply method

2.3.3 set method

2.4 fc_vfemp1.PDE object

This object is used to create the scalar PDE (1.2) or the vector PDE (1.18):

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

Its main properties are

Properties of PDE object	
dim	: integer, space dimension.
d	: integer the PDE acts on d -dimensional surfaces.
m	: integer, number of PDE's.
Op	: Loperator or Hoperator object. If m = 1 , then Op is an Loperator object. Otherwise Op is an Hoperator object with dimension m .
f	: (cells of) Fdata object or empty. Used to store the right-hand side of the PDE. If Op is an Loperator object then f is an Fdata object or is empty. If Op is an Hoperator object then f is a cell array of Op.m Fdata object or empty value.

Its constructor are

```
obj=fc_vfemp1.PDE()
obj=fc_vfemp1.PDE(Op)
obj=fc_vfemp1.PDE(Op, f)
```

Description

`obj=fc_vfemp1.PDE()` create an empty object.

`obj=fc_vfemp1.PDE(Op)` create the PDE with $f \equiv 0$: i.e. $Op(u)=0$

`obj=fc_vfemp1.PDE(Op,f)` create the PDE $Op(u)=f$. If **Op** is an **Hoperator** object then **f** must be a cell array of length **Hoperator.m**.

Example

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

```
Lop=fc_vfemp1.Loperator(2,2,{1,[],[];[],1},[],[],1);
f=@(x,y) x.*sin(x+y);
pde=fc_vfemp1.PDE(Lop, f);
```

The **f** function must be written in a vectorized form.

2.5 fc_vfemp1.BVP object

The object **BVP** is used to create a scalar boundary value problem (1.2)-(1.4) or a vector boundary value problem (1.18)-(1.20). The usage of this object is strongly correlated with good comprehension of the toolbox and and more particularly with the **siMesh** object.

The properties of the object **BVP** are

Properties of BVP object	
<code>dim</code>	: integer, space dimension.
<code>d</code>	: integer the BVP acts on <code>d</code> -dimensional surfaces.
<code>m</code>	: integer, system of <code>m</code> PDEs.
<code>Th</code>	: a <code>siMesh</code> object We must have <code>Th.dim = dim</code> and <code>Th.d = d</code> .
<code>pdes</code>	: <code>Th.nsTh</code> -by-1 cell array. Used to store the PDE associated with each submesh <code>Th.sTh{i}</code> . If <code>pdes{i}</code> is empty then there is no PDE defined on <code>Th.sTh{i}</code> .

2.5.1 Constructor

Its constructor are

```
obj=fc_vfemp1.BVP()
obj=fc_vfemp1.BVP(Th,pde)
obj=fc_vfemp1.BVP(Th,pde,labels)
```

Description

`obj=fc_vfemp1.BVP()` create an empty BVP object.

`obj=fc_vfemp1.BVP(Th,pde)` create a BVP object with PDE's defined by `pde` object on all submeshes of index `Th.find(pde.d)` i.e. on all submeshes such that `Th.sThi==pde.d`. By default, homogeneous Neumann boundary conditions are set on all *boundaries*.

`obj=fc_vfemp1.BVP(Th,pde,labels)` similar to previous one except among the selected objects are chosen those with label (`Th.sTh{i}.label`) in `labels` array. By default, homogeneous Neumann boundary conditions are set on all *boundaries*.

2.5.2 setPDE method

```
obj.setPDE(d,label,pde)
```

2.5.3 Description

`obj.setPDE(d,label,pde)` associated the `pde` object with the `i`-th submesh such that `i=obj.Th.find(d,label)`. If `i` exists then `obj.pdes{i}` is set to `pde`.

2.5.4 setDirichlet method

```
obj.setDirichlet(label,g)
obj.setDirichlet(label,g,Lm)
```

Description

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

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

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

$$u_i = g\{i\}, \forall i \in \llbracket 1, m \rrbracket \text{ on } \Gamma_{\text{label}}.$$

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

$$u_{\text{Lm}(i)} = g\{i\}, \forall i \in \llbracket 1, \text{length}(\text{Lm}) \rrbracket \text{ on } \Gamma_{\text{label}}.$$

2.5.5 setRobin method

```
obj.setRobin(label,gr,ar)
obj.setRobin(label,gr,ar,Lm)
```

Description

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

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

For vector B.V.P., sets Robin boundary condition (1.20)

$$\frac{\partial \mathbf{u}}{\partial n_{\mathcal{H}_i}} + \text{ar}\{i\}.\mathbf{u}_i = \text{gr}\{i\}, \forall i \in \llbracket 1, m \rrbracket \text{ on } \Gamma_{\text{label}}.$$

`obj.setRobin(label,gr,ar,Lm)` for vector B.V.P., sets Robin boundary condition (1.20) :

$\forall i \in \llbracket 1, \text{length}(\text{Lm}) \rrbracket$, let $\alpha = \text{Lm}(i)$ then

$$\frac{\partial \mathbf{u}}{\partial n_{\mathcal{H}_\alpha}} + \text{ar}\{i\}.\mathbf{u}_\alpha = \text{gr}\{i\}, \text{ on } \Gamma_{\text{label}}.$$

2.5.6 solve method

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

Description

`x=obj.solve()` uses \mathbb{P}^1 -Lagrange finite elements method to solve the B.V.P. described by the `bvp` object.

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

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

2.5.7 Assembly method

```
[A,b]=obj.Assembly()
[A,b]=obj.Assembly(key,value,...)
```

Description

`[A,b]=obj.Assembly()` returns the matrix **A** and the vector **b** obtain when applying a P_1 -Lagrange finite elements method on the B.V.P. described by the `obj` object.

`[A,b]=obj.Assembly(key,value,...)`

- 'local' :
- 'physical' :
- 'interface' :
- 'Robin' :
- 'Dirichlet' :
- 'dom' :

Chapter 3

Scalar boundary value problems

3.1 Poisson BVP's

The generic problem to solve is the following

Usual BVP 2 : Poisson problem

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

$$-\Delta u = f \text{ in } \Omega \subset \mathbb{R}^{\dim}, \quad (3.1)$$

$$u = g_D \text{ on } \Gamma_D, \quad (3.2)$$

$$\frac{\partial u}{\partial n} + a_R u = g_R \text{ on } \Gamma_R, \quad (3.3)$$

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

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

$$-\Delta \stackrel{\text{def}}{=} -\sum_{i=1}^{\dim} \frac{\partial^2}{\partial x_i^2} = \mathcal{L}_{1, \mathbf{0}, \mathbf{0}, 0}. \quad (3.4)$$

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

$$\frac{\partial u}{\partial n_{\mathcal{L}}} \stackrel{\text{def}}{=} \langle \mathbb{A} \nabla u, \mathbf{n} \rangle - \langle \mathbf{b}u, \mathbf{n} \rangle = \frac{\partial u}{\partial n}. \quad (3.5)$$

We now will see how to implement different Poisson's BVP while using the toolbox.

3.1.1 2D Poisson BVP with Dirichlet boundary conditions on the unit square

Let Ω be the unit square. The associated mesh can be obtained from

- the `fc_simesh.hypercube` function (see [2] for explanation and Figure 3.1 for graphic representations) by the command

```
Th=fc_simesh.hypercube(2,N);
```

- the `gmsh` software by using the `square4.geo` file (see [1] for explanation and Figure 3.1 for graphic representations) and the commands

```
fullgeofile=fc_vfemp1.get_geo(2,2,'square4');
meshfile=fc_oogmsh.buildmesh2d(fullgeofile,N);
Th=fc_simesh.siMesh(meshfile);
```

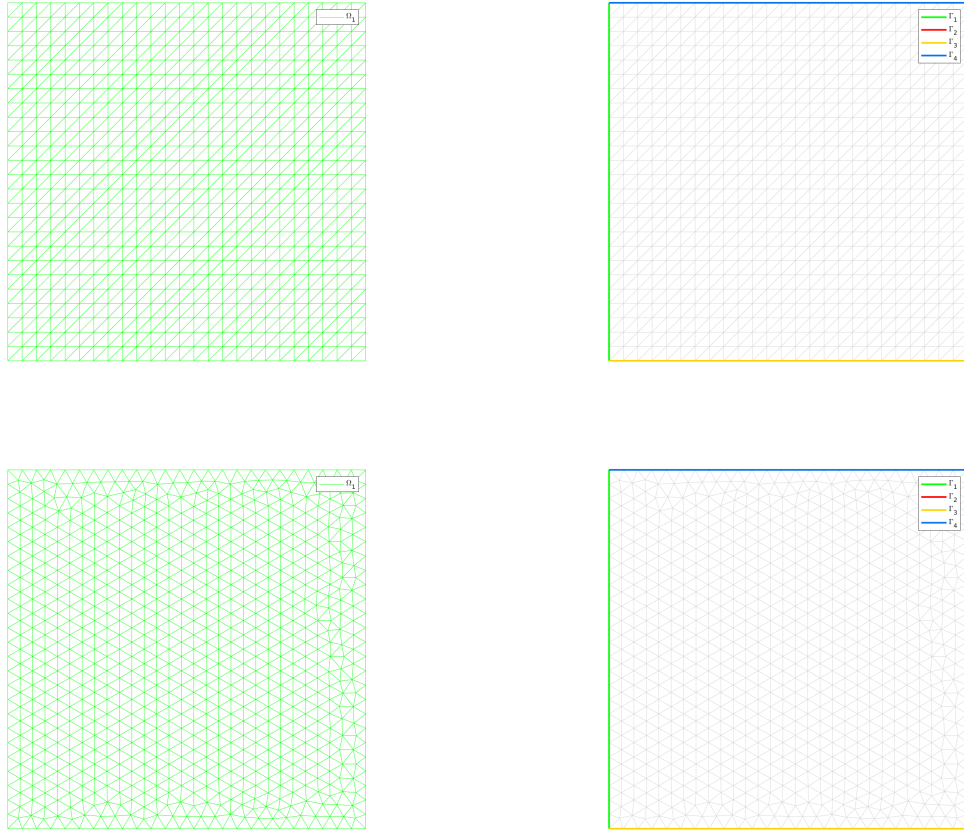


Figure 3.1: Meshes of the unit square (left) and its boundaries (right) generated with the `fc_simesh.hypercube` function (top) and with the `gmsh` software and the `square4.geo` file (bottom)

We choose the problem to have the following exact solution

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

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

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

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

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

So this problem can be written as the scalar BVP 1

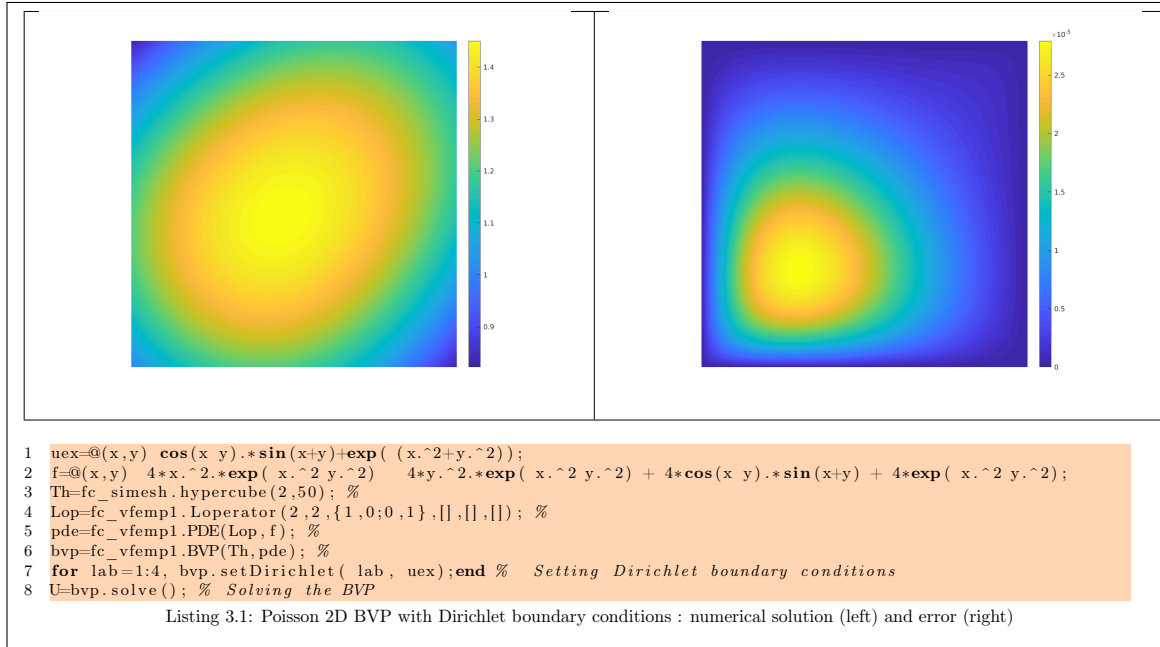
Scalar BVP 3 : 2D Poisson BVP with Dirichlet boundary conditions

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

$$\mathcal{L}_{1,0,0,0}(u) = f \text{ in } \Omega = [0, 1]^2, \quad (3.6)$$

$$u = u_{\text{ex}} \text{ on } \Gamma_1 \cup \Gamma_2 \cup \Gamma_3 \cup \Gamma_4, \quad (3.7)$$

In Listing 3.1, we give the complete code to solve this problem with the toolbox.



There are description of lines:

- line 3: generates a `siMesh` object representing the unit square.
- line 4: generates the `Loperator` object defined in 3.6.
- line 5: generates the `PDE` object representing 3.6.
- line 6: generates the `BVP` object given in *Scalar BVP 3*.
- line 7: sets the `BVP` object boundary conditions to be the dirichlet boundary conditions 3.7.
- line 8: solves the boundary value problem described by the `BVP` object.

Computational times for assembling and solving steps are given in Table 3.1 and Table 3.2 for meshes generated respectively with the `fc_simesh.hypercube` function and with the `gmsh` software. In both cases, the relative errors between exact solution of the 2D Poisson BVP, described in *Scalar BVP 3*, and the numerical P^1 -Lagrange finite element solution are given in Table 3.3 and Table 3.4. At least, in Figure 3.2 orders of the P^1 -Lagrange finite element method are represented: a superconvergence phenomena is observed with the H^1 -norm on the regular mesh generated by the `fc_simesh.hypercube` function.

N	n_q	n_{me}	Assembly	Solve
100	10 201	20 000	0.123 (s)	0.029 (s)
200	40 401	80 000	0.177 (s)	0.127 (s)
300	90 601	180 000	0.410 (s)	0.382 (s)
400	160 801	320 000	0.740 (s)	0.694 (s)
500	251 001	500 000	1.154 (s)	1.144 (s)
600	361 201	720 000	1.698 (s)	1.794 (s)
700	491 401	980 000	2.320 (s)	2.837 (s)
800	641 601	1 280 000	3.236 (s)	4.014 (s)

Table 3.1: Computational times for assembling and solving the 2D Poisson BVP, described in *Scalar BVP 3*, where meshes are generated with the `fc_simesh.hypercube` function.

N	n_q		n_{me}		Assembly	Solve
100	11	827	23	252	0.142 (s)	0.059 (s)
200	46	681	92	560	0.272 (s)	0.284 (s)
300	104	707	208	212	0.634 (s)	0.664 (s)
400	185	703	369	804	1.175 (s)	1.465 (s)
500	290	158	578	314	2.064 (s)	2.427 (s)
600	417	242	832	082	3.019 (s)	3.680 (s)
700	567	287	1 131	772	4.378 (s)	5.960 (s)
800	741	022	1 478	842	5.869 (s)	7.628 (s)

Table 3.2: Computational times for assembling and solving the 2D Poisson BVP, described in *Scalar BVP 3*, where meshes are generated with the `gmsh` software and the `square4.geo` file.

N	n_q		n_{me}		h	L^∞ -error	L^2 -error	H^1 -error
100	10	201	20	000	1.414e-02	2.851e-06	1.324e-06	6.648e-06
200	40	401	80	000	7.071e-03	7.129e-07	3.310e-07	1.662e-06
300	90	601	180	000	4.714e-03	3.168e-07	1.471e-07	7.388e-07
400	160	801	320	000	3.536e-03	1.782e-07	8.276e-08	4.156e-07
500	251	001	500	000	2.828e-03	1.141e-07	5.296e-08	2.660e-07
600	361	201	720	000	2.357e-03	7.921e-08	3.678e-08	1.847e-07
700	491	401	980	000	2.020e-03	5.819e-08	2.702e-08	1.357e-07
800	641	601	1 280	000	1.768e-03	4.455e-08	2.068e-08	1.039e-07

Table 3.3: Relative errors between exact solution of the 2D Poisson BVP, described in *Scalar BVP 3*, and the numerical P^1 -Lagrange finite element solution on the meshes generated with the `fc_simesh.hypercube` function.

N	n_q		n_{me}		h	L^∞ -error	L^2 -error	H^1 -error
100	11	827	23	252	1.273e-02	5.038e-06	1.774e-06	8.302e-05
200	46	681	92	560	6.508e-03	1.269e-06	4.436e-07	4.212e-05
300	104	707	208	212	4.399e-03	7.998e-07	1.965e-07	1.994e-05
400	185	703	369	804	3.252e-03	3.004e-07	1.100e-07	9.828e-06
500	290	158	578	314	2.588e-03	1.932e-07	7.046e-08	8.401e-06
600	417	242	832	082	2.172e-03	1.556e-07	4.893e-08	7.726e-06
700	567	287	1 131	772	1.910e-03	1.593e-07	3.591e-08	4.637e-06
800	741	022	1 478	842	1.663e-03	7.707e-08	2.747e-08	3.432e-06

Table 3.4: Relative errors between exact solution of the 2D Poisson BVP, described in *Scalar BVP 3*, and the numerical P^1 -Lagrange finite element solution on meshes generated with the `gmsh` software and the `square4.geo` file.

remark 3.1

In the `vfemP1` Matlab toolbox some functions were provided to solve the *Scalar BVP 3*.

- the main script solving the BVP with the unit square generated with the `fc_simesh.hypercube` function:

```
fc_vfemp1.examples.Poisson.BVPPoisson2D_ex01
```

- the command for building the BVP with the unit square generated with the `fc_simesh.hypercube` function

```
[bvp,info]=fc_vfemp1.examples.Poisson.setBVPPoisson2D_ex01(N,verbose)
```

or with the unit square generated with the `gmsh` software and the `square4.geo` file:

```
[bvp,info]=fc_vfemp1.examples.Poisson.setBVPPoisson2D_ex01(N,verbose,'hypercube',false)
```

- the commands to run the benchmarks with the `fc_simesh.hypercube` function

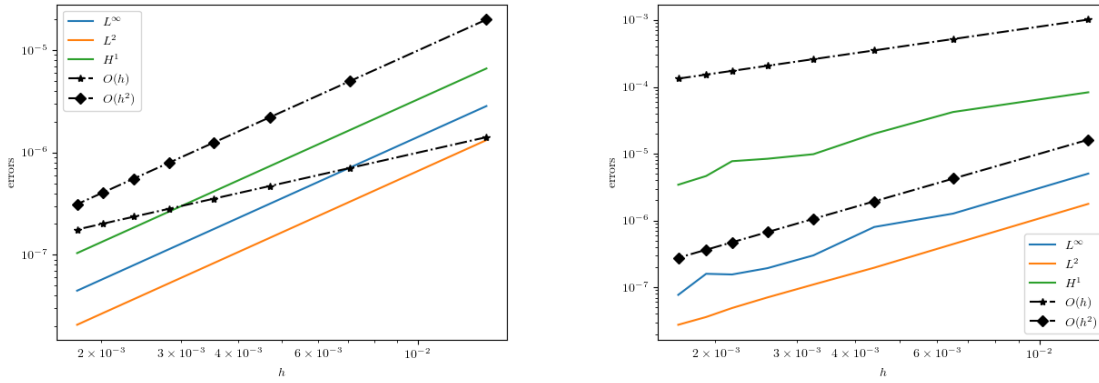


Figure 3.2: 2D Poisson BVP with dirichlet boundary conditions: order of the P_1 -Lagrange finite element method in function of the mesh size h for meshes generated respectively with the `fc_simesh.hypercube` function (left) and with the `gmsh` software (right).

```
setBVP=@(N,verb)fc_vfemp1.examples.Poisson.setBVPpoisson2D_ex01(N,verb);
fc_vfemp1.examples.bench('LN',100:100:800,'setBVP',setBVP);
or with the gmsh software and the square4.geo file:
setBVP=@(N,V)fc_vfemp1.examples.Poisson.setBVPpoisson2D_ex01(N,V,'HyperCube',false);
fc_vfemp1.examples.bench('LN',100:100:800,'setBVP',setBVP);
```

3.1.2 2D Poisson BVP with mixed boundary conditions

Let Ω be the unit square with the associated mesh obtain from `HYPERCUBE` function (see section 3.1.1 for explanation and Figure 3.1 for a mesh sample) or by using the `gmsh` software with the `square4.geo` file.

We choose the problem to have the exact solution given by

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

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

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

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

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

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

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

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

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

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

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

So this problem can be written as the scalar BVP 1

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

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

$$\mathcal{L}_{1,0,0,0}(u) = f \text{ in } \Omega = [0,1]^2, \quad (3.8)$$

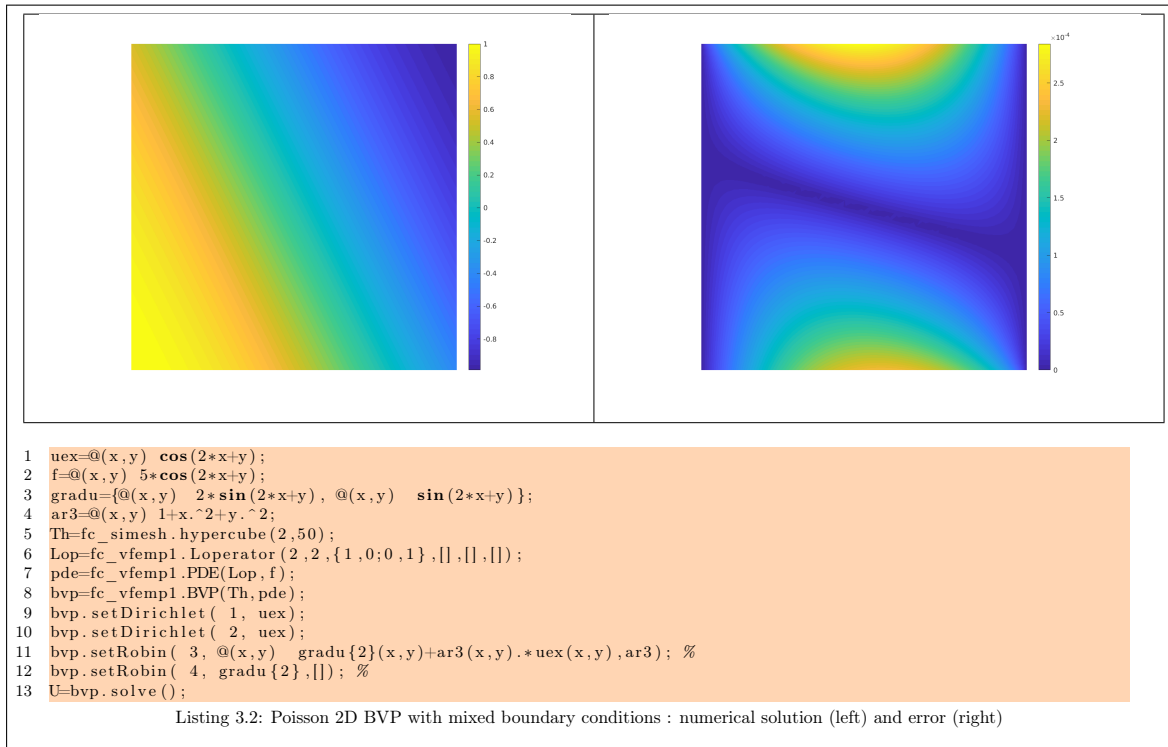
$$u = u_{\text{ex}} \text{ on } \Gamma_1 \cup \Gamma_2 \cup \Gamma_3 \cup \Gamma_4, \quad (3.9)$$

$$\frac{\partial u}{\partial n_{\mathcal{L}}} + a^R u = g^R \text{ on } \Gamma_3, \quad (3.10)$$

$$\frac{\partial u}{\partial n_{\mathcal{L}}} = g^N \text{ on } \Gamma_4, \quad (3.11)$$

$$(3.12)$$

In Listing 3.2, we give the complete code to solve this problem with the toolbox.



We set respectively in lines 11 and 12, the Robin and the Neumann boundary conditions by using `setRobin` member function of `BVP` class.

Computational times for assembling and solving steps are given in Table 3.5 and Table 3.6 for meshes generated respectively with the `fc_simesh.hypercube` function and with the `gmsh` software. In both cases, the relative errors between exact solution of the 2D Poisson BVP, described in *Scalar BVP 4*, and the numerical $Pk1$ -Lagrange finite element solution are given in Table 3.7 and Table 3.8. At least, in Figure 3.3 orders of the $Pk1$ -Lagrange finite element method are represented: a superconvergence phenomena is observed with the H^1 -norm on the regular mesh generated by the `fc_simesh.hypercube` function.

N	n_q		n_{me}		Assembly	Solve
100	10	201	20	000	0.141 (s)	0.029 (s)
200	40	401	80	000	0.193 (s)	0.127 (s)
300	90	601	180	000	0.412 (s)	0.377 (s)
400	160	801	320	000	0.693 (s)	0.690 (s)
500	251	001	500	000	1.120 (s)	1.145 (s)
600	361	201	720	000	1.677 (s)	1.816 (s)
700	491	401	980	000	2.455 (s)	2.796 (s)
800	641	601	1 280	000	3.158 (s)	3.963 (s)

Table 3.5: Computational times for assembling and solving the 2D Poisson BVP, described in *Scalar* BVP 4, where meshes are generated with the `fc_simesh.hypercube` function.

N	n_q		n_{me}		Assembly	Solve
100	11	827	23	252	0.158 (s)	0.058 (s)
200	46	681	92	560	0.278 (s)	0.313 (s)
300	104	707	208	212	0.616 (s)	0.711 (s)
400	185	703	369	804	1.135 (s)	1.334 (s)
500	290	158	578	314	1.910 (s)	2.462 (s)
600	417	242	832	082	2.836 (s)	3.762 (s)
700	567	287	1 131	772	4.025 (s)	5.485 (s)
800	741	022	1 478	842	5.491 (s)	7.662 (s)

Table 3.6: Computational times for assembling and solving the 2D Poisson BVP, described in *Scalar* BVP 4, where meshes are generated with the `gmsh` software and the `square4.geo` file.

N	n_q		n_{me}		h	L^∞ -error	L^2 -error	H^1 -error
100	10	201	20	000	1.414e-02	3.553e-05	1.660e-05	4.578e-05
200	40	401	80	000	7.071e-03	8.884e-06	4.151e-06	1.145e-05
300	90	601	180	000	4.714e-03	3.949e-06	1.845e-06	5.088e-06
400	160	801	320	000	3.536e-03	2.221e-06	1.038e-06	2.862e-06
500	251	001	500	000	2.828e-03	1.422e-06	6.642e-07	1.832e-06
600	361	201	720	000	2.357e-03	9.872e-07	4.612e-07	1.272e-06
700	491	401	980	000	2.020e-03	7.253e-07	3.389e-07	9.345e-07
800	641	601	1 280	000	1.768e-03	5.553e-07	2.594e-07	7.155e-07

Table 3.7: Relative errors between exact solution of the 2D Poisson BVP, described in *Scalar* BVP 4, and the numerical $Pk1$ -Lagrange finite element solution on the meshes generated with the `fc_simesh.hypercube` function.

N	n_q		n_{me}		h	L^∞ -error	L^2 -error	H^1 -error
100	11	827	23	252	1.273e-02	1.222e-05	5.894e-06	1.016e-04
200	46	681	92	560	6.508e-03	3.953e-06	1.464e-06	4.482e-05
300	104	707	208	212	4.399e-03	1.878e-06	6.500e-07	2.198e-05
400	185	703	369	804	3.252e-03	7.498e-07	3.656e-07	1.255e-05
500	290	158	578	314	2.588e-03	6.105e-07	2.342e-07	9.366e-06
600	417	242	832	082	2.172e-03	4.975e-07	1.625e-07	8.327e-06
700	567	287	1 131	772	1.910e-03	3.636e-07	1.193e-07	5.617e-06
800	741	022	1 478	842	1.663e-03	1.864e-07	9.135e-08	4.408e-06

Table 3.8: Relative errors between exact solution of the 2D Poisson BVP, described in *Scalar* BVP 4, and the numerical $Pk1$ -Lagrange finite element solution on meshes generated with the `gmsh` software and the `square4.geo` file.

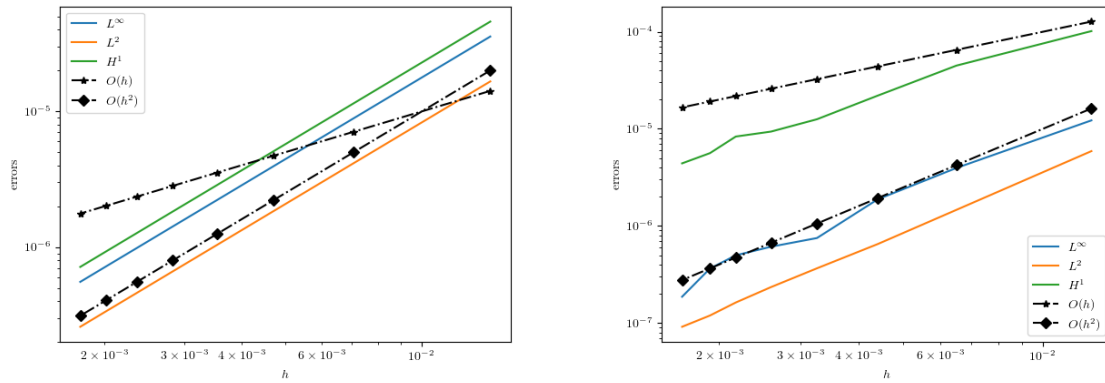


Figure 3.3: 2D Poisson BVP with mixed boundary conditions: order of the P_1 -Lagrange finite element method in function of the mesh size h for meshes generated respectively with the `fc_simesh.hypercube` function (left) and with the `gmsh` software (right).

remark 3.2

In the `vfemp1` Matlab toolbox some functions were provided to solve the *Scalar BVP 4*.

- the main script solving the BVP with the unit square generated with the `fc_simesh.hypercube` function:

```
fc_vfemp1.examples.Poisson.BVPPoisson2D_ex03
```

- the command for building the BVP with the unit square generated with the `fc_simesh.hypercube` function

```
[bvp,info]=fc_vfemp1.examples.Poisson.setBVPPoisson2D_ex03(N,verbose)
```

or with the unit square generated with the `gmsh` software and the `square4.geo` file:

```
[bvp,info]=fc_vfemp1.examples.Poisson.setBVPPoisson2D_ex03(N,verbose,'hypercube',false)
```

- the commands to run the benchmarks with the `fc_simesh.hypercube` function

```
setBVP=@(N,verb)fc_vfemp1.examples.Poisson.setBVPPoisson2D_ex03(N,verb);
fc_vfemp1.examples.bench('LN',100:100:800,'setBVP',setBVP);
```

or with the `gmsh` software and the `square4.geo` file:

```
setBVP=@(N,V)fc_vfemp1.examples.Poisson.setBVPPoisson2D_ex03(N,V,'hypercube',false);
fc_vfemp1.examples.bench('LN',100:100:800,'setBVP',setBVP);
```

3.1.3 3D Poisson BVP with mixed boundary conditions

Let Ω be the unit cube. The associated mesh can be obtained from

- the `fc_simesh.hypercube` function (see [2] for explanation and Figure 3.4 for graphic representations) by the command

```
Th=fc_simesh.hypercube(3,N);
```

- the `gmsh` software by using the `cube6.geo` file (see [1] for explanation and Figure 3.1 for graphic representations) and the commands

```
fullgeofile=fc_vfemp1.get_geo(3,3,'cube6');
meshfile=fc_oogmsh.buildmesh3d(fullgeofile,N);
Th=fc_simesh.siMesh(meshfile);
```

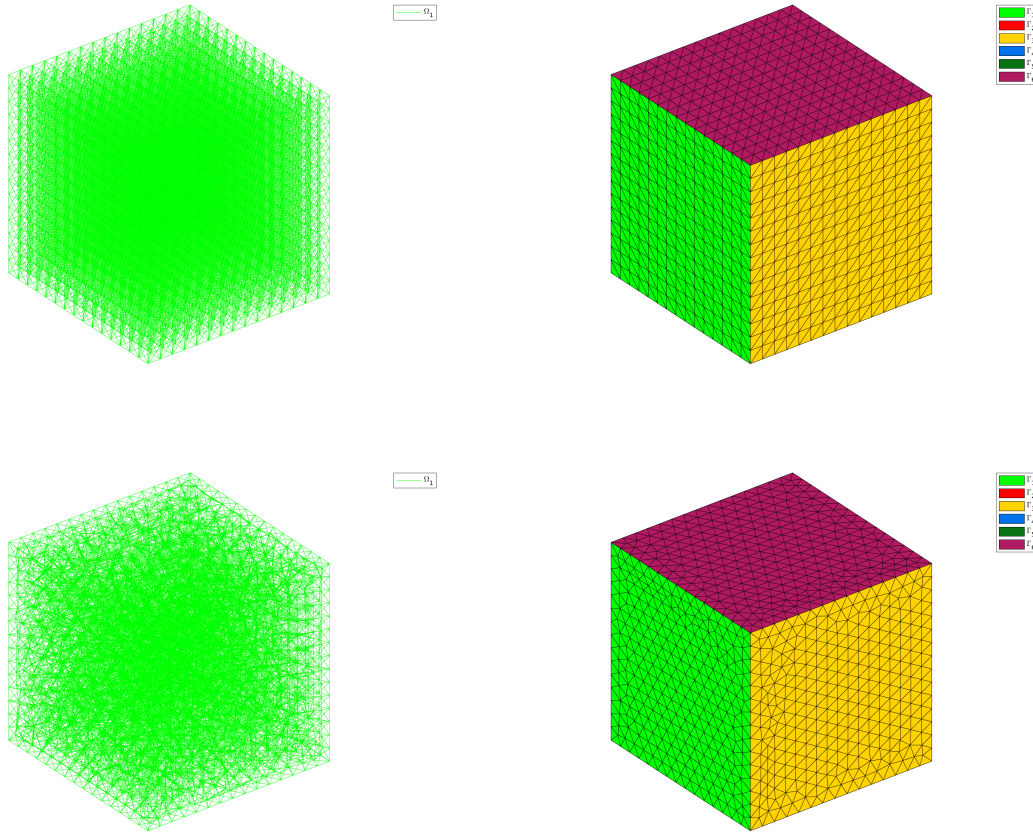


Figure 3.4: Meshes of the unit cube (left) and its boundaries (right) generated with the `fc_simesh.hypercube` function (top) and with the `gmsh` software and the `cube6.geo` file (bottom)

We choose the problem to have exact solution

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

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

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

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

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

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

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

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

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

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

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

So this problem can be written as the scalar BVP 1

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

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

$$\mathcal{L}_{1,0,0,0}(u) = f \text{ in } \Omega = [0, 1]^3, \quad (3.13)$$

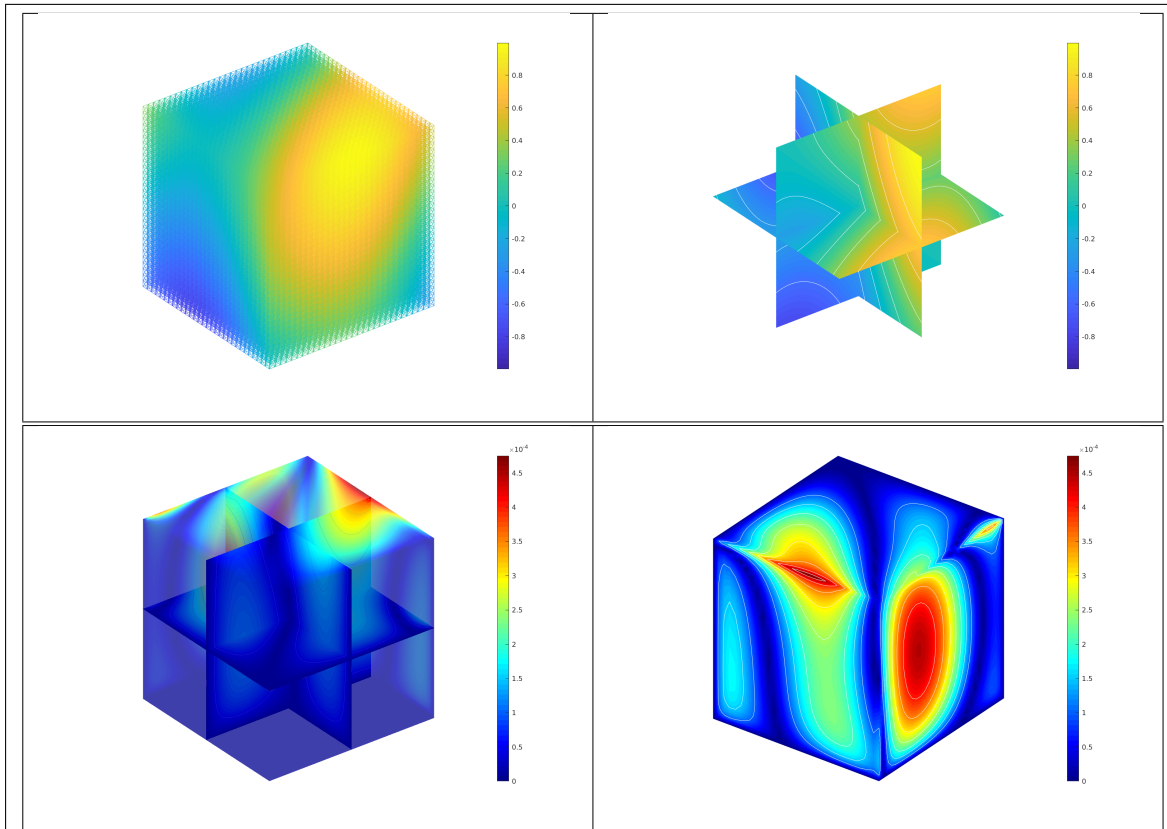
$$u = u_{\text{ex}} \text{ on } \Gamma_1 \cup \Gamma_3 \cup \Gamma_5, \quad (3.14)$$

$$\frac{\partial u}{\partial n_{\mathcal{L}}} + a^R u = g^R \text{ on } \Gamma_2 \cup \Gamma_4, \quad (3.15)$$

$$\frac{\partial u}{\partial n_{\mathcal{L}}} = g^N \text{ on } \Gamma_6, \quad (3.16)$$

$$(3.17)$$

In Listing 3.3, we give the complete code to solve this problem with the toolbox.



```

uex=@(x,y,z) cos(2*x-y-z).*sin(x+2*y+z);
f=@(x,y,z) 6*cos(x-2*y+z).*sin(2*x-y-z) + 12*cos(2*x-y-z).*sin(x-2*y+z);
ar=1;
gradu=@(x,y,z) cos(2*x-y-z).*cos(x+2*y+z) - 2*sin(2*x-y-z).*sin(x+2*y+z), ...
@(x,y,z) 2*cos(2*x-y-z).*cos(x+2*y+z) + sin(2*x-y-z).*sin(x+2*y+z), ...
@(x,y,z) cos(2*x-y-z).*cos(x+2*y+z) + sin(2*x-y-z).*sin(x+2*y+z);
Th=fc_simesh.hypercube(3,30);
Lop=fc_vfempl.Loperator(3,3,{1,0,0;0,1,0;0,0,1},[],[],[]);
pde=fc_vfempl.PDE(Lop,f);
bvp=fc_vfempl.BVP(Th,pde);
for lab=[1,3,5], bvp.setDirichlet(lab,uex);end
bvp.setRobin(2,@(x,y,z) gradu{1}(x,y,z)+ar*uex(x,y,z),ar);
bvp.setRobin(4,@(x,y,z) gradu{2}(x,y,z)+ar*uex(x,y,z),ar);
bvp.setRobin(6,@(x,y,z) gradu{3}(x,y,z),[]);
U=bvp.solve();

```

Listing 3.3: 3D Poisson BVP with mixed boundary conditions : numerical solution (upper) and error (bottom)

Computational times for assembling and solving steps are given in Table 3.9 and Table 3.10 for meshes generated respectively with the `fc_simesh.hypercube` function and with the `gmsh` software. In both cases, the relative errors between exact solution of the 3D Poisson BVP, described in *Scalar BVP 5*, and the numerical $Pk1$ -Lagrange finite element solution are given in Table 3.11 and Table 3.12. At least, in Figure 3.5 orders of the $Pk1$ -Lagrange finite element method are represented: a superconvergence phenomena is observed with the H^1 -norm on the regular mesh generated by the `fc_simesh.hypercube` function.

N	n_q	n_{me}	Assembly	Solve
20	9 261	48 000	0.334 (s)	0.309 (s)
25	17 576	93 750	0.444 (s)	1.211 (s)
30	29 791	162 000	0.762 (s)	2.364 (s)
35	46 656	257 250	1.197 (s)	6.434 (s)
40	68 921	384 000	1.827 (s)	15.507 (s)
45	97 336	546 750	2.643 (s)	25.157 (s)
50	132 651	750 000	3.582 (s)	45.440 (s)
55	175 616	998 250	4.735 (s)	57.406 (s)

Table 3.9: Computational times for assembling and solving the 3D Poisson BVP, described in *Scalar BVP 5*, where meshes are generated with the `fc_simesh.hypercube` function.

N	n_q	n_{me}	Assembly	Solve
20	7 321	36 470	0.314 (s)	0.217 (s)
25	13 769	71 726	0.403 (s)	0.475 (s)
30	22 817	122 187	0.685 (s)	0.975 (s)
35	35 162	192 730	1.064 (s)	1.974 (s)
40	51 568	287 339	1.658 (s)	3.687 (s)
45	72 570	409 908	2.373 (s)	7.239 (s)
50	98 013	559 094	3.381 (s)	10.699 (s)
55	129 523	745 396	4.558 (s)	19.155 (s)

Table 3.10: Computational times for assembling and solving the 3D Poisson BVP, described in *Scalar BVP 5*, where meshes are generated with the `gmsh` software and the `square4.geo` file.

N	n_q	n_{me}	h	L^∞ -error	L^2 -error	H^1 -error
20	9 261	48 000	8.660e-02	1.205e-02	4.203e-03	9.966e-03
25	17 576	93 750	6.928e-02	8.537e-03	2.700e-03	6.416e-03
30	29 791	162 000	5.774e-02	6.388e-03	1.878e-03	4.475e-03
35	46 656	257 250	4.949e-02	4.984e-03	1.382e-03	3.299e-03
40	68 921	384 000	4.330e-02	4.008e-03	1.059e-03	2.532e-03
45	97 336	546 750	3.849e-02	3.301e-03	8.369e-04	2.006e-03
50	132 651	750 000	3.464e-02	2.771e-03	6.781e-04	1.628e-03
55	175 616	998 250	3.149e-02	2.363e-03	5.606e-04	1.348e-03

Table 3.11: Relative errors between exact solution of the 3D Poisson BVP, described in *Scalar BVP 5*, and the numerical $Pk1$ -Lagrange finite element solution on the meshes generated with the `fc_simesh.hypercube` function.

N	n_q	n_{me}	h	L^∞ -error	L^2 -error	H^1 -error
20	7 321	36 470	1.099e-01	2.585e-02	5.242e-03	3.094e-02
25	13 769	71 726	8.375e-02	1.451e-02	3.287e-03	2.413e-02
30	22 817	122 187	7.550e-02	1.048e-02	2.302e-03	2.004e-02
35	35 162	192 730	6.320e-02	9.019e-03	1.696e-03	1.689e-02
40	51 568	287 339	5.384e-02	6.944e-03	1.282e-03	1.448e-02
45	72 570	409 908	4.829e-02	5.349e-03	1.020e-03	1.293e-02
50	98 013	559 094	4.471e-02	5.137e-03	8.268e-04	1.147e-02
55	129 523	745 396	4.001e-02	3.755e-03	6.817e-04	1.047e-02

Table 3.12: Relative errors between exact solution of the 3D Poisson BVP, described in *Scalar BVP 5*, and the numerical $Pk1$ -Lagrange finite element solution on meshes generated with the `gmsh` software and the `square4.geo` file.

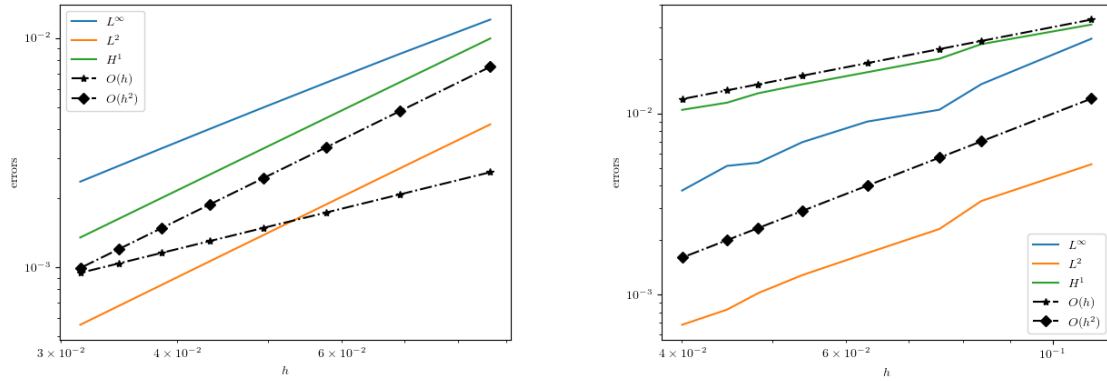


Figure 3.5: 3D Poisson BVP with mixed boundary conditions: order of the P_1 -Lagrange finite element method in function of the mesh size h for meshes generated respectively with the `fc_simesh.hypercube` function (left) and with the `gmsh` software (right).

remark 3.3

In the `cvfemP1` Matlab toolbox some functions were provided to solve the *Scalar* BVP 5.

- the main script solving the BVP with the unit square generated with the `fc_simesh.hypercube` function:

```
fc_vfemp1.examples.Poisson.BVPPoisson3D_ex01
```

- the command for building the BVP with the unit square generated with the `fc_simesh.hypercube` function:

```
[bvp,info]=fc_vfemp1.examples.Poisson.setBVPPoisson3D_ex01(N,verbose)
```

or with the unit square generated with the `gmsh` software and the `cube6.geo` file:

```
[bvp,info]=fc_vfemp1.examples.Poisson.setBVPPoisson3D_ex01(N,verbose,'hypercube',false)
```

- the commands to run the benchmarks with the `fc_simesh.hypercube` function:

```
setBVP=@(N,verb)fc_vfemp1.examples.Poisson.setBVPPoisson3D_ex01(N,verb);
```

```
fc_vfemp1.examples.bench('LN',20:5:60,'setBVP',setBVP);
```

or with the `gmsh` software and the `cube6.geo` file:

```
setBVP=@(N,V)fc_vfemp1.examples.Poisson.setBVPPoisson3D_ex01(N,V,'hypercube',false);
```

```
fc_vfemp1.examples.bench('LN',20:5:60,'setBVP',setBVP);
```

3.1.4 4D Poisson BVP with mixed boundary conditions

Let Ω be the unit 4D-hypercube. The associated mesh can be obtained from the `fc_simesh.hypercube` function (see [2] for explanation by the command

```
Th=fc_simesh.hypercube(4,N);
```

We choose the problem to have exact solution

$$u_{\text{ex}}(\mathbf{x}) = \cos(2x_1 - x_2 - x_3 + x_4) \sin(x_1 - 2x_2 + x_3 - 2x_4).$$

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

$$f(\mathbf{x}) = 2 \cos(x_1 - 2x_2 + x_3 - 2x_4) \sin(2x_1 - x_2 - x_3 + x_4) \\ + 17 \cos(2x_1 - x_2 - x_3 + x_4) \sin(x_1 - 2x_2 + x_3 - 2x_4).$$

On boundary labels 1, 3, 5, 7 we set a Dirichlet boundary conditions :

$$u = u_{\text{ex}}, \quad \text{on } \Gamma^D = \Gamma_1 \cup \Gamma_3 \cup \Gamma_5 \cup \Gamma_7.$$

On boundary labels 2, 4, we choose a Robin boundary condition with $a^R(\mathbf{x}) = 1$. So we have

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

with

$$g^R(\mathbf{x}) = \cos(2x_1 - x_2 - x_3 + x_4) \cos(x_1 - 2x_2 + x_3 - 2x_4) + \cos(2x_1 - x_2 - x_3 + x_4) \sin(x_1 - 2x_2 + x_3 - 2x_4) - 2 \sin(2x_1 - x_2 - x_3 + x_4) \sin(x_1 - 2x_2 + x_3 - 2x_4), \quad \forall \mathbf{x} \in \Gamma_2$$

and

$$g^R(\mathbf{x}) = -2 \cos(2x_1 - x_2 - x_3 + x_4) \cos(x_1 - 2x_2 + x_3 - 2x_4) + \cos(2x_1 - x_2 - x_3 + x_4) \sin(x_1 - 2x_2 + x_3 - 2x_4) + \sin(2x_1 - x_2 - x_3 + x_4) \sin(x_1 - 2x_2 + x_3 - 2x_4), \quad \forall \mathbf{x} \in \Gamma_4.$$

On boundary labels 6, 8, we choose a Neumann boundary condition. So we have

$$\frac{\partial u}{\partial n} = g^N, \text{ on } \Gamma^N = \Gamma_6 \cup \Gamma_8$$


with

$$g^N(\mathbf{x}) = \cos(2x_1 - x_2 - x_3 + x_4) \cos(x_1 - 2x_2 + x_3 - 2x_4) + \sin(2x_1 - x_2 - x_3 + x_4) \sin(x_1 - 2x_2 + x_3 - 2x_4), \quad \forall \mathbf{x} \in \Gamma_6$$

and

$$g^N(\mathbf{x}) = -2 \cos(2x_1 - x_2 - x_3 + x_4) \cos(x_1 - 2x_2 + x_3 - 2x_4) - \sin(2x_1 - x_2 - x_3 + x_4) \sin(x_1 - 2x_2 + x_3 - 2x_4), \quad \forall \mathbf{x} \in \Gamma_8.$$

The Neumann boundary conditions can also be written as Robin boundary conditions with $a^R = 0$ on $\Gamma_6 \cup \Gamma_8$. So this problem can be written as the scalar BVP 1

 **Scalar BVP 6 : 4D Poisson BVP with mixed boundary conditions**

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

$$\mathcal{L}_{1,0,0,0}(u) = f \text{ in } \Omega = [0, 1]^4, \quad (3.18)$$

$$u = u_{\text{ex}} \text{ on } \Gamma_1 \cup \Gamma_3 \cup \Gamma_5 \cup \Gamma_7, \quad (3.19)$$

$$\frac{\partial u}{\partial n_{\mathcal{L}}} + a^R u = g^R \text{ on } \Gamma_2 \cup \Gamma_4, \quad (3.20)$$

$$\frac{\partial u}{\partial n_{\mathcal{L}}} = g^N \text{ on } \Gamma_6 \cup \Gamma_8, \quad (3.21)$$

$$(3.22)$$

In Listing 3.4, we give the complete code to solve this problem with the toolbox.

```

uex=@(x1,x2,x3,x4) cos(2*x1-x2-x3+x4)*sin(x1-2*x2+x3-2*x4);
f=@(x1,x2,x3,x4) 2*cos(x1-2*x2+x3-2*x4)*sin(2*x1-x2-x3+x4)+17*cos(2*x1-x2-x3+x4)*sin(x1-2*x2+x3-2*x4);
ar=1;
gradu=@(x1,x2,x3,x4) cos(2*x1-x2-x3+x4)*cos(x1-2*x2+x3-2*x4)+sin(2*x1-x2-x3+x4)*sin(x1-2*x2+x3-2*x4);
gradu=@(x1,x2,x3,x4) 2*cos(2*x1-x2-x3+x4)*cos(x1-2*x2+x3-2*x4)+sin(2*x1-x2-x3+x4)*sin(x1-2*x2+x3-2*x4);
gradu=@(x1,x2,x3,x4) cos(2*x1-x2-x3+x4)*cos(x1-2*x2+x3-2*x4)+sin(2*x1-x2-x3+x4)*sin(x1-2*x2+x3-2*x4);
gradu=@(x1,x2,x3,x4) 2*cos(2*x1-x2-x3+x4)*cos(x1-2*x2+x3-2*x4)+sin(2*x1-x2-x3+x4)*sin(x1-2*x2+x3-2*x4);
Th=fc_simesh.hypercube(4,10);
Lop=fc_vfempl.Loperator(4,4,{1,0,0,0;0,1,0,0;0,0,1,0;0,0,0,1},[],[],[]);
pde=fc_vfempl.PDE(Lop,f);
bvp=fc_vfempl.BVP(Th,pde);
for lab=[1,3,5,7], bvp.setDirichlet(lab,uex);end
bvp.setRobin(2,@(x1,x2,x3,x4) gradu{1}(x1,x2,x3,x4)+ar*uex(x1,x2,x3,x4),ar);
bvp.setRobin(4,@(x1,x2,x3,x4) gradu{2}(x1,x2,x3,x4)+ar*uex(x1,x2,x3,x4),ar);
bvp.setRobin(6,@(x1,x2,x3,x4) gradu{3}(x1,x2,x3,x4),[]);
bvp.setRobin(8,@(x1,x2,x3,x4) gradu{4}(x1,x2,x3,x4),[]);
U=bvp.solve();

```

Listing 3.4: 4D Poisson BVP with mixed boundary conditions

Computational times for assembling and solving steps are given in Table 3.13. The relative errors between exact solution of the 4D Poisson BVP, described in *Scalar BVP 6*, and the numerical $Pk1$ -Lagrange finite element solution are given in Table 3.14. At least, in Figure 3.6 orders of the $Pk1$ -Lagrange finite element method are represented: a superconvergence phenomena is observed with the H^1 -norm on the regular mesh generated by the `fc_simesh.hypercube` function.

N	n_q	n_{me}	Assembly	Solve
5	1 296	15 000	0.274 (s)	0.037 (s)
7	4 096	57 624	0.522 (s)	0.411 (s)
9	10 000	157 464	1.471 (s)	2.447 (s)
11	20 736	351 384	3.146 (s)	14.253 (s)
13	38 416	685 464	6.115 (s)	100.844 (s)

Table 3.13: Computational times for assembling and solving the 4D Poisson BVP, described in *Scalar BVP 6*, where meshes are generated with the `fc_simesh.hypercube` function.

N	n_q	n_{me}	h	L^∞ -error	L^2 -error	H^1 -error
5	1 296	15 000	4.000e-01	2.256e-02	6.989e-03	2.462e-02
7	4 096	57 624	2.857e-01	1.312e-02	3.755e-03	1.344e-02
9	10 000	157 464	2.222e-01	1.035e-02	2.323e-03	8.426e-03
11	20 736	351 384	1.818e-01	8.278e-03	1.574e-03	5.768e-03
13	38 416	685 464	1.538e-01	6.641e-03	1.135e-03	4.195e-03

Table 3.14: Relative errors between exact solution of the 4D Poisson BVP, described in *Scalar BVP 6*, and the numerical $Pk1$ -Lagrange finite element solution on the meshes generated with the `fc_simesh.hypercube` function.

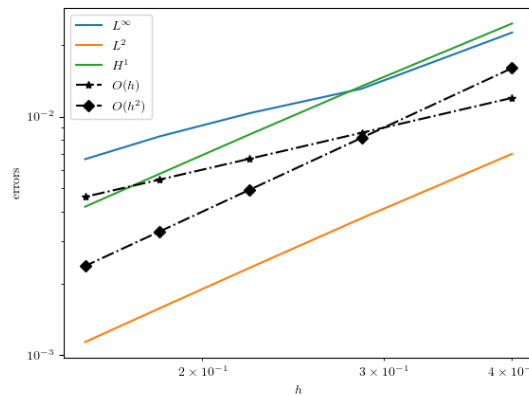


Figure 3.6: 4D Poisson BVP with mixed boundary conditions: order of the P_1 -Lagrange finite element method in function of the mesh size h for meshes generated with the `fc_simesh.hypercube` function.

remark 3.4

In the `vfemP1` Matlab toolbox some functions were provided to solve the *Scalar BVP 6*.

- the main script solving the BVP with the unit square generated with the `fc_simesh.hypercube` function:

```
fc_vfemP1.examples.Poisson.BVPPoisson4D_ex01
```

- the command for building the BVP with the unit square generated with the `fc_simesh.hypercube` function:

```
[bvp,info]=fc_vfemp1.examples.Poisson.setBVPPoisson4D_ex01(N,verbose)
```

- the commands to run the benchmarks with the `fc_simesh.hypercube` function:

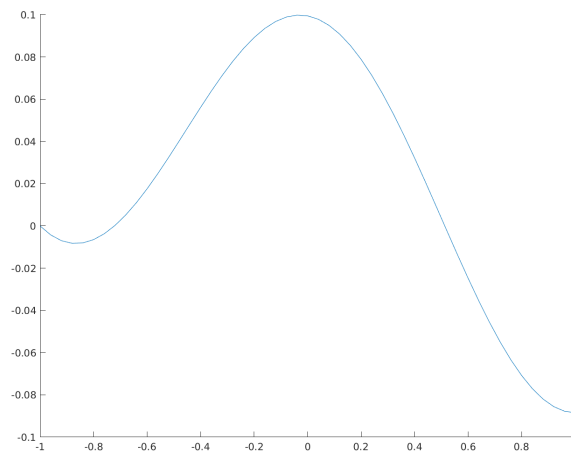
```
setBVP=@(N,verb)fc_vfemp1.examples.Poisson.setBVPPoisson4D_ex01(N,verb);
fc_vfemp1.examples.bench('LN',5:2:15,'setBVP',setBVP);
```

3.1.5 1D BVP : just for fun

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

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

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



```
f=@(x) cos(pi*x);
c=@(x) 1+(x-1).^2;
a=1;b=1;
Th=fc_simesh.hypercube(1,50,'trans',@(x) a + (b-a)*x);
Lop=fc_vfemp1.Loperator(1,1,{1},[],[],c);
pde=fc_vfemp1.PDE(Lop,f);
bvp=fc_vfemp1.BVP(Th,pde);
bvp.setDirichlet(1,0);
U=bvp.solve();
figure(1)
Th.plot(U)
```

Listing 3.5: 1D BVP with mixed boundary conditions

3.2 Stationary convection-diffusion problem

3.2.1 Stationary convection-diffusion problem in 2D

The 2D problem to solve is the following

Usual BVP 3 : 2D stationary convection-diffusion problem

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

$$-\operatorname{div}(\alpha \nabla u) + \langle \mathbf{V}, \nabla u \rangle + \beta u = f \quad \text{in } \Omega \subset \mathbb{R}^2, \quad (3.23)$$

$$u = 4 \quad \text{on } \Gamma_2, \quad (3.24)$$

$$u = -4 \quad \text{on } \Gamma_4, \quad (3.25)$$

$$u = 0 \quad \text{on } \Gamma_{20} \cup \Gamma_{21}, \quad (3.26)$$

$$\frac{\partial u}{\partial n} = 0 \quad \text{on } \Gamma_1 \cup \Gamma_3 \cup \Gamma_{10} \quad (3.27)$$

where Ω and its boundaries are given in Figure 3.7. This problem is well posed if $\alpha(\mathbf{x}) > 0$ and $\beta(\mathbf{x}) \geq 0$. We choose α , \mathbf{V} , β and f in Ω as :

$$\alpha(\mathbf{x}) = 0.1 + (x_1 - 0.5)^2,$$

$$\mathbf{V}(\mathbf{x}) = (-10x_2, 10x_1)^t,$$

$$\beta(\mathbf{x}) = 0.01,$$

$$f(\mathbf{x}) = -200 \exp(-10((x_1 - 0.75)^2 + x_2^2)).$$

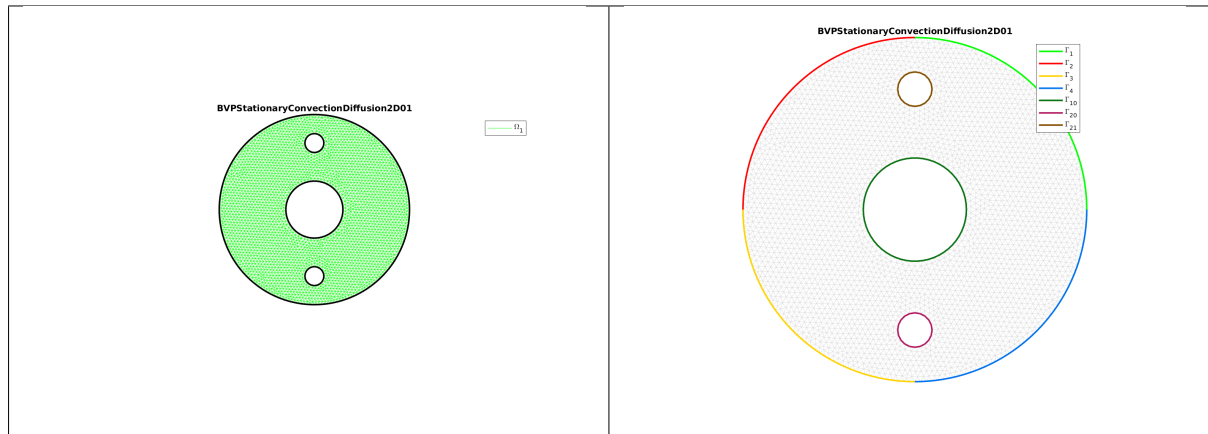


Figure 3.7: 2D stationary convection-diffusion BVP : mesh (left) and boundaries (right)

The problem (3.23)-(3.27) can be equivalently expressed as the scalar BVP (1.2)-(1.4) :

Scalar BVP 7 : 2D stationary convection-diffusion problem

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

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

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

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

where

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

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

- $\Gamma^D = \Gamma_2 \cup \Gamma_4 \cup \Gamma_{20} \cup \Gamma_{21}$ 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 $\Gamma_{20} \cup \Gamma_{21}$

$$\bullet a^R = g^R := 0 \text{ on } \Gamma^R.$$

The algorithm using the toolbox for solving (3.23)-(3.27) is the following:

Algorithm 1 Stationary convection-diffusion problem in 2D

```

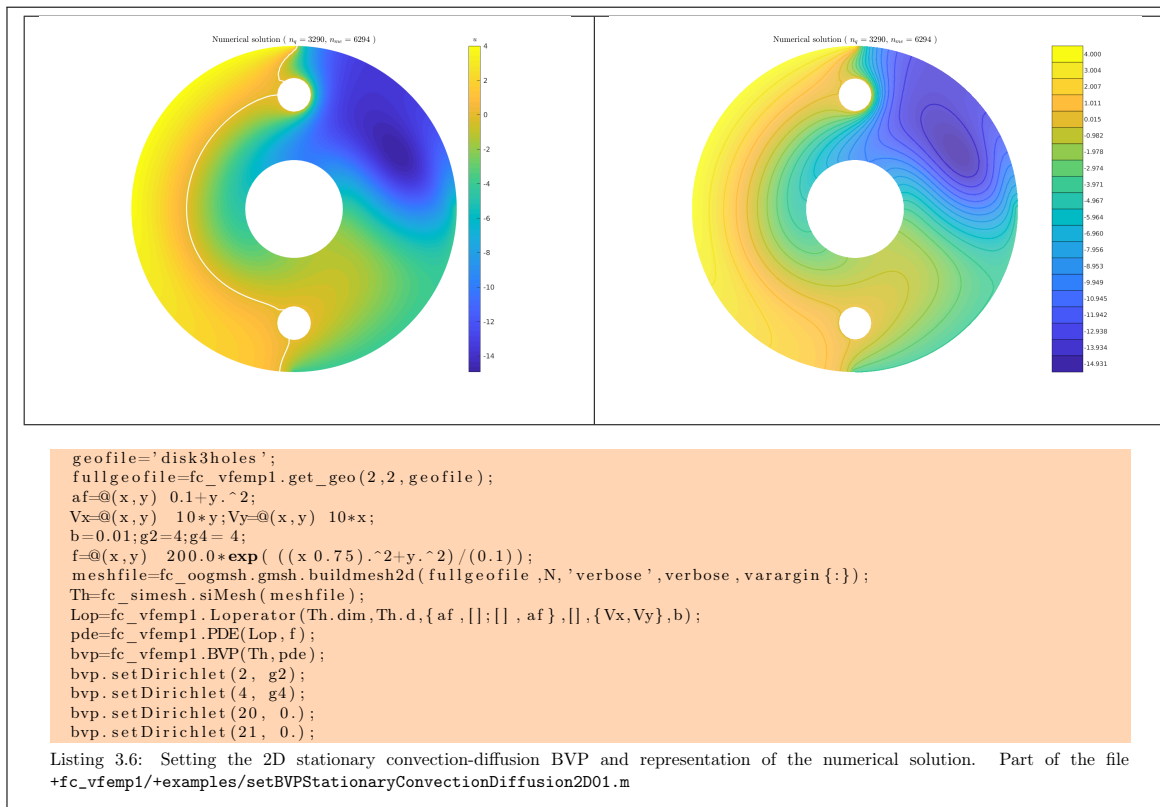
1:  $\mathcal{T}_h \leftarrow \text{SIMESH}(\dots)$  ▷ Get mesh
2:  $\alpha \leftarrow (x, y) \mapsto 0.1 + (y - 0.5)(y - 0.5)$ 
3:  $\beta \leftarrow 0.01$ 
4:  $f \leftarrow (x, y) \mapsto -200e^{-10((x-0.75)^2+y^2)}$ 
5:  $\text{Lop} \leftarrow \text{LOPERATOR}(2, 2, \begin{pmatrix} \alpha & 0 \\ 0 & \alpha \end{pmatrix}, \mathbf{0}, \begin{pmatrix} -10y \\ 10x \end{pmatrix}, \beta)$ 
6:  $\text{pde} \leftarrow \text{PDE}(\text{Lop}, f)$ 
7:  $\text{bvp} \leftarrow \text{BVP}(\mathcal{T}_h, \text{pde})$ 
8:  $\text{bvp.SETDIRICHLET}(2, 4.0)$  ▷ Set 'Dirichlet' condition on  $\Gamma_2$ 
9:  $\text{bvp.SETDIRICHLET}(4, -4.0)$  ▷ Set 'Dirichlet' condition on  $\Gamma_4$ 
10:  $\text{bvp.SETDIRICHLET}(20, 0.0)$  ▷ Set 'Dirichlet' condition on  $\Gamma_{20}$ 
11:  $\text{bvp.SETDIRICHLET}(21, 0.0)$  ▷ Set 'Dirichlet' condition on  $\Gamma_{21}$ 
12:  $\mathbf{u} \leftarrow \text{bvp.SOLVE}()$ 

```

In Listing 3.6 the code to set the *Scalar* BVP 7 is given and the numerical solution obtained by using the command `u=bvp.solve()` is represented. The complete code, with graphical representations, is provided in the toolbox by the script:

`+fc_vfemp1/+examples/BVPStationaryConvectionDiffusion2D01.m`

In Table 3.15 computational times for assembling and solving steps are given with various size meshes.



N	n_q		n_{me}		Assembly	Solve
100	33	503	66	060	0.275 (s)	0.194 (s)
200	131	165	260	444	0.835 (s)	0.899 (s)
300	294	356	585	882	2.089 (s)	2.303 (s)
400	521	219	1 038	668	3.869 (s)	4.519 (s)
500	814	835	1 624	954	6.497 (s)	7.677 (s)
600	1 169	219	2 332	782	9.802 (s)	11.872 (s)

Table 3.15: Computational times for assembling and solving the 2D stationary convection-diffusion BVP, described in *Scalar BVP 7*, with various size meshes.

remark 3.5

In the `fvemp1` Matlab toolbox some functions were provided to solve the *Scalar BVP 7*.

- the main script solving the BVP, and with graphic representations:
`fc_vfemp1.examples.BVPStationaryConvectionDiffusion2D01`
- the function for building the BVP with the mesh file:
`[bvp,info]=fc_vfemp1.examples.setBVPStationaryConvectionDiffusion2D01(N,verbose)`
- the commands to run the benchmarks are:

```
setBVP=@(N,verbose) fc_vfemp1.examples.setBVPStationaryConvectionDiffusion2D01(N,verbose);
fc_vfemp1.examples.bench('LN',100:100:600,'setBVP',setBVP);
```

3.2.2 Stationary convection-diffusion problem in 3D

Let $A = (x_A, y_A) \in \mathbb{R}^2$ and $C_A^r([z_{min}, z_{max}])$ 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

$$\Omega = C_{(0,0)}^1([0, 3]) \setminus \{C_{(0,0)}^{0.3}([0, 3]) \cup C_{(0,-0.7)}^{0.1}([0, 3]) \cup C_{(0,0.7)}^{0.1}([0, 3])\}.$$

We respectively denote by Γ_{100} and Γ_{101} the $z = 0$ and $z = 3$ bases of Ω .

Γ_1 , Γ_{10} , Γ_{20} and Γ_{21} are respectively the curved surfaces of cylinders $C_{(0,0)}^1([0, 3])$, $C_{(0,0)}^{0.3}([0, 3])$, $C_{(0,-0.7)}^{0.1}([0, 3])$ and $C_{(0,0.7)}^{0.1}([0, 3])$. The domain Ω and its boundaries are represented in Figure 3.8.

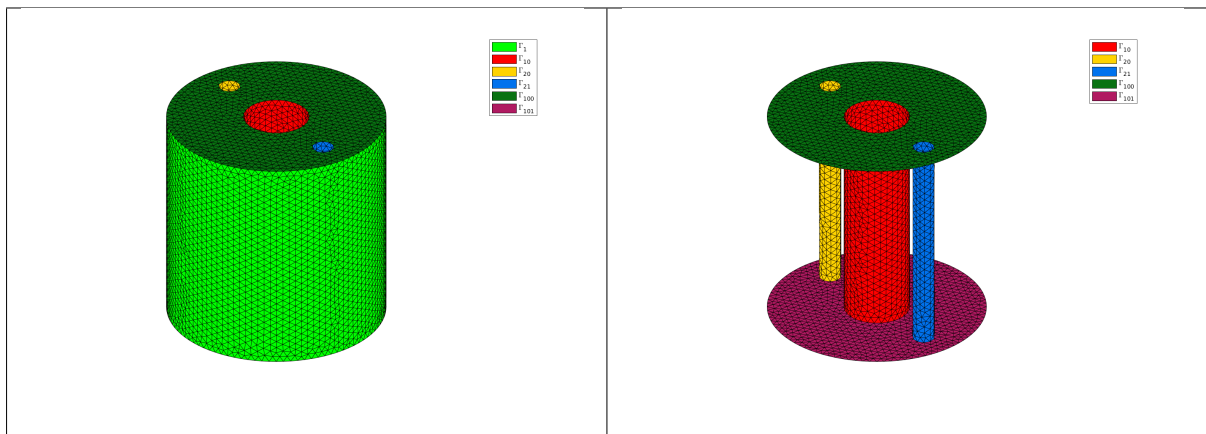



Figure 3.8: 3D stationary convection-diffusion BVP : all boundaries (left) and boundaries without Γ_1 (right)

The 3D problem to solve is the following

 **Usual BVP 4 :**

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

$$-\operatorname{div}(\alpha \nabla u) + \langle \mathbf{V}, \nabla u \rangle + \beta u = f \text{ in } \Omega \subset \mathbb{R}^3, \tag{3.28}$$

$$\alpha \frac{\partial u}{\partial n} + a_{20} u = g_{20} \text{ on } \Gamma_{20}, \tag{3.29}$$


$$\alpha \frac{\partial u}{\partial n} + a_{21} u = g_{21} \text{ on } \Gamma_{21}, \tag{3.30}$$

$$\frac{\partial u}{\partial n} = 0 \text{ on } \Gamma^N \tag{3.31}$$

where $\Gamma^N = \Gamma_1 \cup \Gamma_{10} \cup \Gamma_{100} \cup \Gamma_{101}$. This problem is well posed if $\alpha(\mathbf{x}) > 0$ and $\beta(\mathbf{x}) \geq 0$. We choose $a_{20} = a_{21} = 1$, $g_{21} = -g_{20} = 0.05$ $\beta = 0.01$ and :

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

The problem (3.28)-(3.31) can be equivalently expressed as the scalar BVP (1.2)-(1.4) :

 **Scalar BVP 8 : 3D stationary convection-diffusion problem**

Find $u \in H^2(\Omega)$ 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

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

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

- $\Gamma^R = \Gamma_1 \cup \Gamma_{10} \cup \Gamma_{20} \cup \Gamma_{21} \cup \Gamma_{100} \cup \Gamma_{101}$ (and $\Gamma^D = \emptyset$)
-

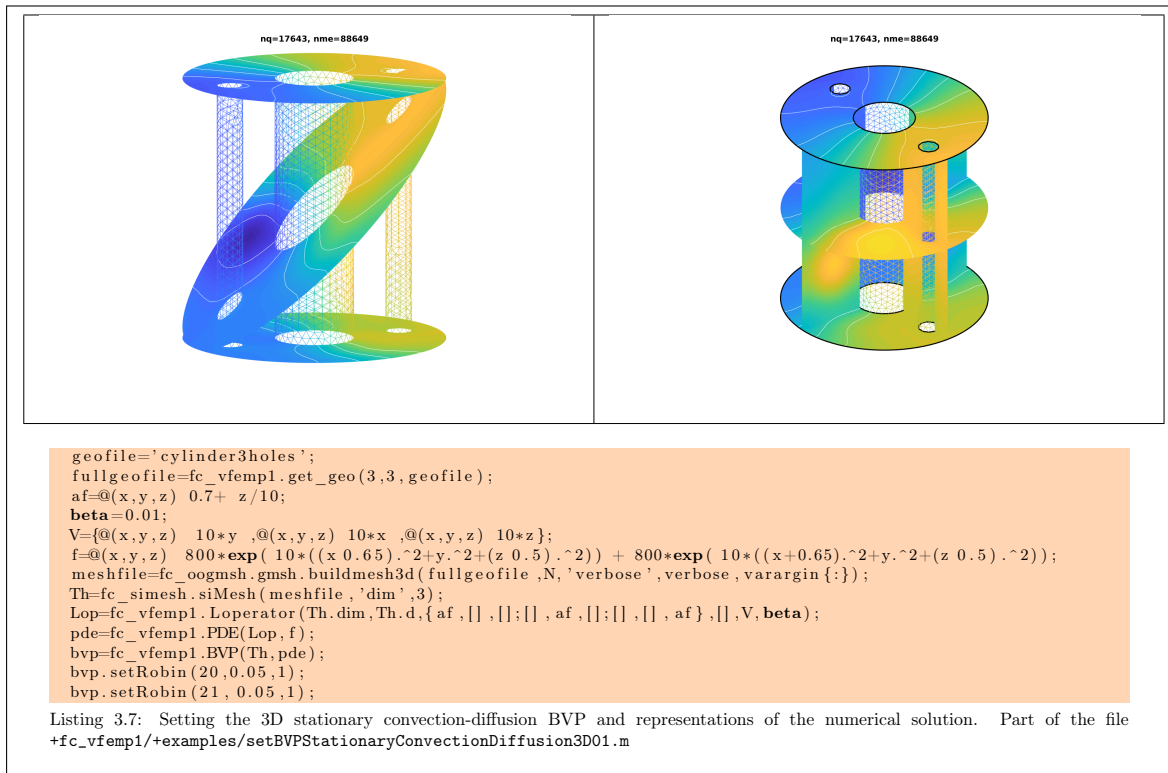
$$a^R = \begin{cases} 0 & \text{on } \Gamma_1 \cup \Gamma_{10} \cup \Gamma_{100} \cup \Gamma_{101} \\ 1 & \text{on } \Gamma_{20} \cup \Gamma_{21} \end{cases}$$

$$g^R = \begin{cases} 0 & \text{on } \Gamma_1 \cup \Gamma_{10} \cup \Gamma_{100} \cup \Gamma_{101} \\ 0.05 & \text{on } \Gamma_{21}, \\ -0.05 & \text{on } \Gamma_{20} \end{cases}$$

In Listing 3.7 the code to set the *Scalar BVP 7* is given and the numerical solution obtained by using the command `u=bvp.solve()` is represented. The complete code, with graphical representations, is provided in the toolbox by the script:

```
+fc_vfemp1/+examples/BVPStationaryConvectionDiffusion3D01.m
```

in Table 3.16 computational times for assembling and solving steps are given with various size meshes.



N	n_q	n_{me}	Assembly	Solve
15	17 643	88 649	0.522 (s)	0.565 (s)
20	38 614	204 301	1.028 (s)	2.184 (s)
25	72 245	395 317	2.060 (s)	4.794 (s)
30	120 802	674 231	3.697 (s)	12.166 (s)
35	186 763	1 061 295	6.021 (s)	27.754 (s)

Table 3.16: Computational times for assembling and solving the 3D stationary convection-diffusion BVP, described in *Scalar BVP 8*, with various size meshes.

remark 3.6

In the `fc_vfemp1` Matlab toolbox some functions were provided to solve the *Scalar BVP 8*.

- the main script solving the BVP, and with graphic representations:
`fc_vfemp1.examples.BVPStationaryConvectionDiffusion3D01`
- the function for building the BVP with the mesh file:
`[bvp,info]=fc_vfemp1.examples.setBVPStationaryConvectionDiffusion3D01(N,verbose)`
- the commands to run the benchmarks are:

```

setBVP=@(N,verbose) fc_vfemp1.examples.setBVPStationaryConvectionDiffusion3D01(N,verbose);
fc_vfemp1.examples.bench('LN',15:5:35,'setBVP',setBVP);

```

3.3 2D electrostatic BVPs

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

$$\operatorname{div} \mathbf{j} = 0, \quad \text{in } \Omega. \quad (3.32)$$

$$\mathbf{j} = \sigma \mathbf{E}, \text{ in } \Omega. \tag{3.33}$$

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

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

$$\mathbf{E} = -\nabla \varphi, \text{ in } \Omega. \tag{3.34}$$

Combining all these equations leads to Laplace's equation

$$\operatorname{div}(\sigma \nabla \varphi) = 0 \tag{3.35}$$

In the resistive model, a good conductor has high value of σ and a good insulator has $0 < \sigma^{-1}$.

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

As example, we use the mesh obtain with gmsh from `square4holes6dom.geo` file represented in Figure 3.9

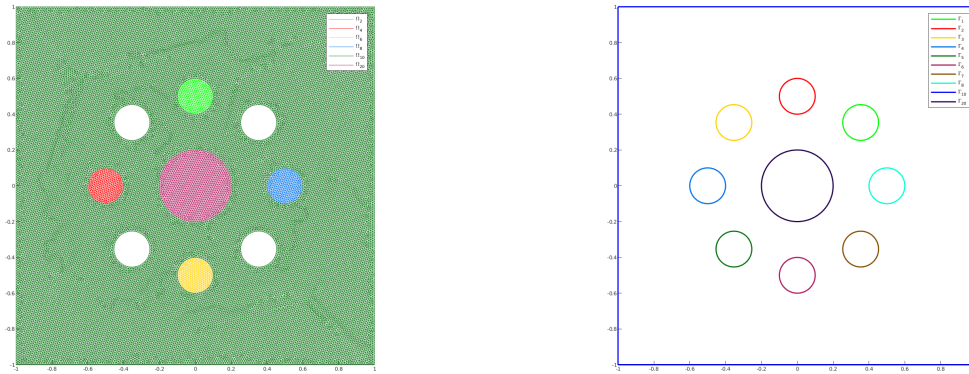


Figure 3.9: Mesh from `square4holes6dom.geo`, domains representation (left) and boundaries (right)

We have two resistive medias

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

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

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

We solve the following BVP

Usual BVP 5 : 2D electrostatic problem

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

$\operatorname{div}(\sigma \nabla \varphi) = 0$	in $\Omega,$	(3.36)
$\varphi = 0$	on $\Gamma_3 \cup \Gamma_7,$	(3.37)
$\varphi = 12$	on $\Gamma_1 \cup \Gamma_5,$	(3.38)
$\sigma \frac{\partial \varphi}{\partial n} = 0$	on $\Gamma_{10}.$	(3.39)

The problem (3.36)-(3.39) can be equivalently expressed as the scalar BVP (1.2)-(1.4) :

Scalar BVP 9 : 2D electrostatic problem

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

$$\begin{aligned} \mathcal{L}(\varphi) &= 0 && \text{in } \Omega, \\ \varphi &= g^D && \text{on } \Gamma^D, \\ \frac{\partial \varphi}{\partial n_{\mathcal{L}}} + a^R \varphi &= g^R && \text{on } \Gamma^R. \end{aligned}$$

where

- $\mathcal{L} := \mathcal{L}_{\sigma_1, \mathbf{0}, \mathbf{V}, \beta}$, and then the conormal derivative of φ is given by

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

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

To write this problem properly with the  toolbox we split (3.36) in two parts

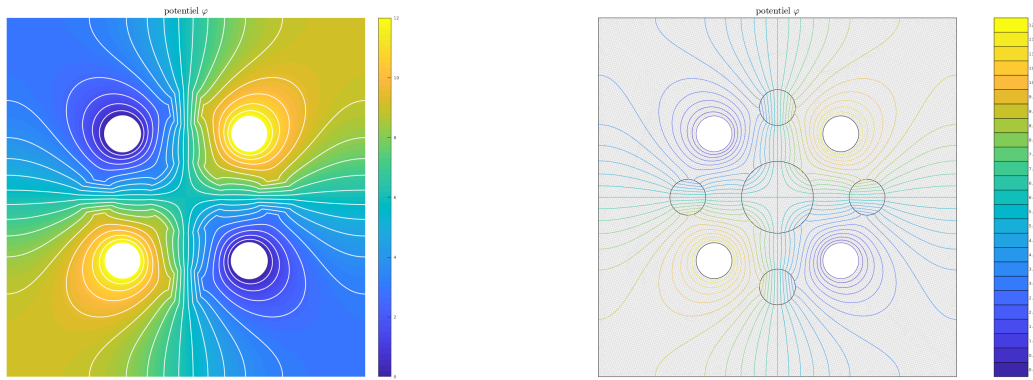
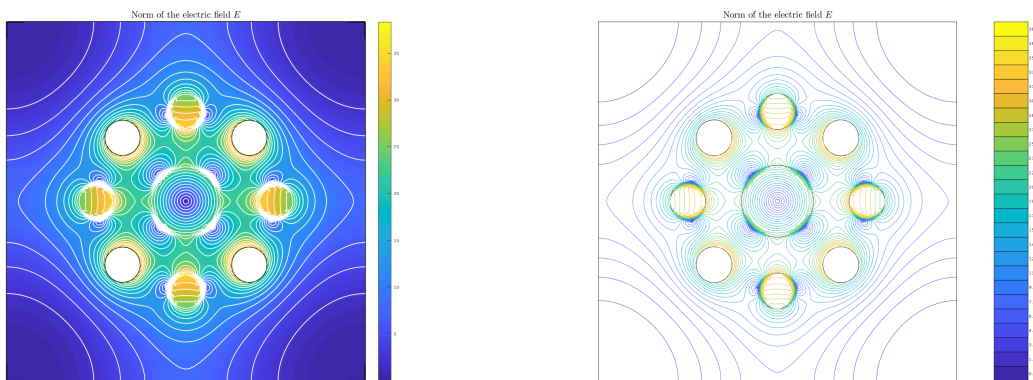
$$\begin{aligned} \operatorname{div}(\sigma_a \nabla \varphi) &= 0 && \text{in } \Omega_a \\ \operatorname{div}(\sigma_b \nabla \varphi) &= 0 && \text{in } \Omega_b \end{aligned}$$

and we set these PDEs on each domains. This is done in Matlab Listing 3.8.

Listing 3.8: **Setting the 2D electrostatic BVP, Matlab code**

```
varargin=fc_tools.utils.deleteCellOptions(varargin,p.Parameters);
sigma1=R.sigma1;sigma2=R.sigma2;%verbose=R.verbose;
geofile='square4holes6dom.geo';
if verbose>=2,fprintf('***_Building_the_mesh_using_GMSH\n\n'>_from_...
    %s.geo\n',geofile);end
tstart=tic();
end
tstart=tic();
Lop=fc_vfemp1.Loperator(dim,d,{sigma2,0;0,sigma2},[],[],[]);
pde=fc_vfemp1.PDE(Lop);
bvp=fc_vfemp1.BVP(Th,pde);
Lop=fc_vfemp1.Loperator(dim,d,{sigma1,0;0,sigma1},[],[],[]);
pde=fc_vfemp1.PDE(Lop);
bvp.setPDE(2,10,pde);
bvp.setDirichlet(1,12);
bvp.setDirichlet(3,0);
```

We show in Figures 3.10 and 3.11 respectively the potential φ and the norm of the electric field \mathbf{E} . Computational times for assembling and solving steps are given in Table 3.17.

Figure 3.10: Test 1, potential φ Figure 3.11: Test 1, norm of the electrical field E

N	n_q		n_{me}		Assembly	Solve
100	46	731	92	412	0.238 (s)	0.253 (s)
150	104	604	207	630	0.389 (s)	0.723 (s)
200	184	691	367	276	0.709 (s)	1.288 (s)
250	288	466	574	298	1.183 (s)	2.226 (s)
300	414	577	825	992	1.779 (s)	3.498 (s)

Table 3.17: Computational times for assembling and solving the 2D stationary convection-diffusion BVP, described in *Scalar BVP 9*, with various size meshes.

remark 3.7

In the  Matlab toolbox some functions were provided to solve the *Scalar BVP ??*.

- the main script solving the BVP, and with graphic representations:
`fc_vfemp1.examples.BVPElectrostatic2D01`
- the function for building the BVP with the mesh file:
`[bvp,info]=fc_vfemp1.examples.setBVPElectrostatic2D01(N,verbose)`
- the commands to run the benchmarks are:

```
setBVP=@(N, verbose) fc_vfemp1.examples.setBVPElectrostatic2D01(N, verbose);  
fc_vfemp1.examples.bench('LN', 100:50:300, 'setBVP', setBVP);
```

- Another function provided allows to choose values of σ_1 and σ_2 :
`fc_vfemp1.examples.runBVPElectrostatic2D01(50, 'sigma1', 0.1, 'sigma2', 100)`

Chapter 4

Vector boundary value problems

4.1 Elasticity problem

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

We consider here Hooke's law in linear elasticity, under small strain hypothesis (see for example [5]).

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

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

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

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

where $\underline{\sigma}$ 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

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

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

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

$$\sigma_{\alpha\beta}(\mathbf{u}) = 2\mu \epsilon_{\alpha\beta}(\mathbf{u}) + \lambda \operatorname{tr}(\underline{\epsilon}(\mathbf{u})) \delta_{\alpha\beta} \quad \forall \alpha, \beta \in \llbracket 1, d \rrbracket$$

The problem to solve is the following



Usual vector BVP 2 : Elasticity problem

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

$$-\operatorname{div}(\boldsymbol{\sigma}(\mathbf{u})) = \mathbf{f}, \text{ in } \Omega \subset \mathbb{R}^d, \quad (4.1)$$

$$\boldsymbol{\sigma}(\mathbf{u}) \cdot \mathbf{n} = \mathbf{0} \text{ on } \Gamma^R, \quad (4.2)$$

$$\mathbf{u} = \mathbf{0} \text{ on } \Gamma^D. \quad (4.3)$$

Now, with the following lemma, we obtain that this problem can be rewritten as the vector BVP defined by (1.18) to (1.20).



Lemme 4.1

Let \mathcal{H} be the d -by- d matrix of the second order linear differential operators defined in (1.14) where $\mathcal{H}_{\alpha,\beta} = \mathcal{L}_{\mathbb{A}^{\alpha,\beta}, \mathbf{0}, \mathbf{0}, 0}$, $\forall (\alpha, \beta) \in \llbracket 1, d \rrbracket^2$, with

$$(\mathbb{A}^{\alpha,\beta})_{k,l} = \mu \delta_{\alpha\beta} \delta_{kl} + \mu \delta_{k\beta} \delta_{l\alpha} + \lambda \delta_{k\alpha} \delta_{l\beta}, \quad \forall (k, l) \in \llbracket 1, d \rrbracket^2. \quad (4.4)$$

then

$$\mathcal{H}(\mathbf{u}) = -\operatorname{div} \boldsymbol{\sigma}(\mathbf{u}) \quad (4.5)$$

and, $\forall \alpha \in \llbracket 1, d \rrbracket$,

$$\frac{\partial \mathbf{u}}{\partial n_{\mathcal{H}_\alpha}} = (\boldsymbol{\sigma}(\mathbf{u}) \cdot \mathbf{n})_\alpha. \quad (4.6)$$

The proof is given in appendix 1.1. So we obtain



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

Let \mathcal{H} be the d -by- d matrix of the second order linear differential operators defined in (1.14) where $\forall (\alpha, \beta) \in \llbracket 1, d \rrbracket^2$, $\mathcal{H}_{\alpha,\beta} = \mathcal{L}_{\mathbb{A}^{\alpha,\beta}, \mathbf{0}, \mathbf{0}, 0}$, with

- for $d = 2$,

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

- for $d = 3$,

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

The elasticity problem (4.1) to (4.3) can be rewritten as :
Find $\mathbf{u} = (\mathbf{u}_1, \dots, \mathbf{u}_d) \in (\mathbb{H}^2(\Omega))^d$ such that

$$\mathcal{H}(\mathbf{u}) = \mathbf{f}, \quad \text{in } \Omega, \quad (4.7)$$

$$\frac{\partial \mathbf{u}}{\partial n_{\mathcal{H}_\alpha}} = 0, \quad \text{on } \Gamma_\alpha^R = \Gamma^R, \quad \forall \alpha \in \llbracket 1, d \rrbracket \quad (4.8)$$

$$\mathbf{u}_\alpha = 0, \quad \text{on } \Gamma_\alpha^D = \Gamma^D, \quad \forall \alpha \in \llbracket 1, d \rrbracket. \quad (4.9)$$



Figure 4.1: Domain and boundaries

4.1.2 2D example

For example, in 2d, we want to solve the elasticity problem (4.1) to (4.3) where Ω and its boundaries are given in Figure 4.1. The material's properties are given by Young's modulus E and Poisson's coefficient ν . As we use plane strain hypothesis, Lamé'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 $E = 21 \cdot 10^5 \text{Pa}$ and $\nu = 0.45$. We also have $\mathbf{f} = \mathbf{x} \mapsto (0, -1)^t$ and we choose $\Gamma^R = \Gamma^2 \cup \Gamma^3 \cup \Gamma^4$, $\Gamma^D = \Gamma^1$.



Vector BVP 4 : Elasticity problem in dimension 2

Let \mathcal{H} be the 2-by-2 matrix defined in *Vector BVP 3*. The elasticity problem (4.1) to (4.3) can be rewritten as :

Find $\mathbf{u} = (\mathbf{u}_1, \mathbf{u}_2) \in (\mathbb{H}^2(\Omega))^2$ such that

$$\mathcal{H}(\mathbf{u}) = (0, -1)^t, \quad \text{in } \Omega, \quad (4.10)$$

$$\frac{\partial \mathbf{u}}{\partial n_{\mathcal{H}_\alpha}} = 0, \quad \text{on } \Gamma_\alpha^R = \Gamma^2 \cup \Gamma^3 \cup \Gamma^4, \quad \forall \alpha \in \llbracket 1, 2 \rrbracket \quad (4.11)$$

$$\mathbf{u}_\alpha = 0, \quad \text{on } \Gamma_\alpha^D = \Gamma^1, \quad \forall \alpha \in \llbracket 1, 2 \rrbracket. \quad (4.12)$$

In Listing 4.1, we give the code to set this problem with the toolbox. This code is a part of the `fc_vfemp1.examples.elasticity.setBVPElasticity2D01` function.

Listing 4.1: 2D elasticity, Matlab code

```
dim=2;
mu= E/(2*(1+nu));
lam = E*nu/((1+nu)*(1-2*nu));
Th=fc_simesh.HyperCube(dim,[round(L/2)*N,N], 'trans',@(q) [L*q(1,:); 1+2*q(2,:)]);
lambda=lam;
gamma=lambda+2*mu;
Hop=fc_vfemp1.Hoperator(dim,dim,dim);
Hop.set(1,1,fc_vfemp1.Loperator(dim,dim,{gamma,[],[]},mu,[],[],[]));
Hop.set(1,2,fc_vfemp1.Loperator(dim,dim,{[],lambda;mu,[]},[],[],[]));
Hop.set(2,1,fc_vfemp1.Loperator(dim,dim,{[],mu;lambda,[]},[],[],[]));
Hop.set(2,2,fc_vfemp1.Loperator(dim,dim,{mu,[],[]},gamma,[],[],[]));
f={0,1};
pde=fc_vfemp1.PDE(Hop,f);
bvp=fc_vfemp1.BVP(Th,pde);
bvp.setDirichlet(1,0.,1:2);
```

One can also use the Matlab function `fc_vfemp1.operators.StiffElas` to build the elasticity operator :

```
Hop=fc_vfemp1.operators.StiffElas(dim,lambda,mu);
```

For a given mesh, its displacement scaled by a factor 50 is shown on Figure 4.2. Computational times for assembling and solving steps are given in Table 4.1.

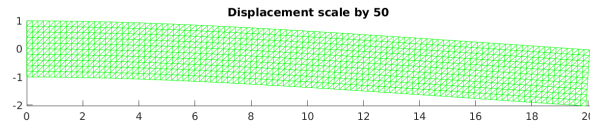


Figure 4.2: Mesh displacement scaled by a factor 50 for the 2D elasticity problem

N	n_q		n_{me}		n_{dof}		Assembly	Solve
100	101	101	200	000	202	202	1.219 (s)	1.621 (s)
150	226	651	450	000	453	302	2.630 (s)	4.701 (s)
200	402	201	800	000	804	402	4.870 (s)	7.289 (s)
250	627	751	1 250	000	1 255	502	8.028 (s)	15.038 (s)
300	903	301	1 800	000	1 806	602	11.625 (s)	50.445 (s)

Table 4.1: Computational times for assembling and solving the 2D elasticity BVP, described in *Vector* BVP 4, with various size meshes.

remark 4.2

In the `vfemP1` Matlab toolbox some codes were provided to solve the *Vector* BVP 4.

- the main script solving the BVP, and with graphic representations:

```
fc_vfemp1.examples.elasticity.BVPElasticity2D01
```

- the function for building the BVP with the mesh file:

```
[bvp,info]=fc_vfemp1.examples.elasticity.setBVPElasticity2D01(N,verbose)
```

- the commands to run the benchmarks are:

```
setBVP=@(N,verbose) fc_vfemp1.examples.elasticity.setBVPElasticity2D01(N,verbose);
fc_vfemp1.examples.bench('LN',100:50:300,'setBVP',setBVP);
```

4.1.3 3D example

Let $\Omega = [0, 5] \times [0, 1] \times [0, 1] \subset \mathbb{R}^3$. The boundary of Ω is made of six faces and each one has a unique label : 1 to 6 respectively for faces $x_1 = 0$, $x_1 = 5$, $x_2 = 0$, $x_2 = 1$, $x_3 = 0$ and $x_3 = 1$. We represent them in Figure 4.3.

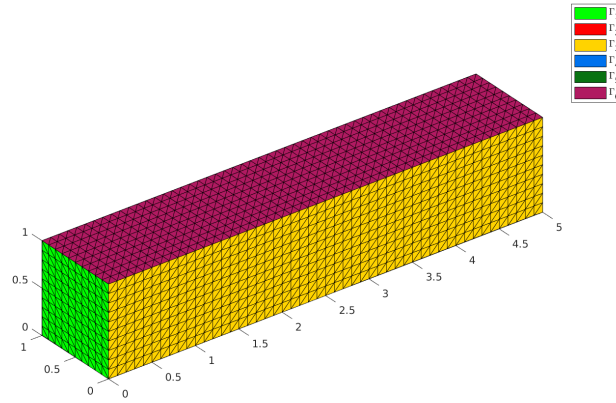


Figure 4.3: Domain and boundaries

We want to solve the elasticity problem (4.1) to (4.3) with $\Gamma^D = \Gamma_1$, $\Gamma^N = \bigcup_{i=2}^6 \Gamma_i$ and $\mathbf{f} = \mathbf{x} \mapsto (0, 0, -1)^t$.

Vector BVP 5 : Elasticity problem in dimension 3

Let \mathcal{H} be the 3-by-3 matrix defined in *Vector BVP 3*. The elasticity problem (4.1) to (4.3) can be rewritten as :

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

$$\mathcal{H}(\mathbf{u}) = (0, 0, -1)^t, \quad \text{in } \Omega, \quad (4.13)$$

$$\frac{\partial \mathbf{u}}{\partial n_{\Gamma_\alpha}} = 0, \quad \text{on } \Gamma_\alpha^R = \bigcup_{i=2}^6 \Gamma_i, \quad \forall \alpha \in \llbracket 1, 3 \rrbracket \quad (4.14)$$

$$\mathbf{u}_\alpha = 0, \quad \text{on } \Gamma_\alpha^D = \Gamma^1, \quad \forall \alpha \in \llbracket 1, 3 \rrbracket. \quad (4.15)$$

In Listing 4.2, we give the code to set this problem with the toolbox. This code is a part of the `fc_vfemp1.examples.elasticity.setBVPElasticity3D01` function.

Listing 4.2: 3D elasticity, Matlab code

```
dim=3;
mu= E/(2*(1+nu));
lam = E*nu/((1+nu)*(1-2*nu));
Th=fc_simesh.HyperCube(dim,[round(L)*N,N,N], 'trans',@(q) [L*q(1,:);q(2,:);q(3,:)]);
Hop=fc_vfemp1.operators.StiffElas(dim,lam,mu);
f={0,0,1};
pde=fc_vfemp1.PDE(Hop,f);
bvp=fc_vfemp1.BVP(Th,pde);
bvp.setDirichlet(1,0.,1:3);
```

The displacement scaled by a factor 2000 for a given mesh is shown on Figure 4.4. Computational times for assembling and solving steps are given in Table 4.2.

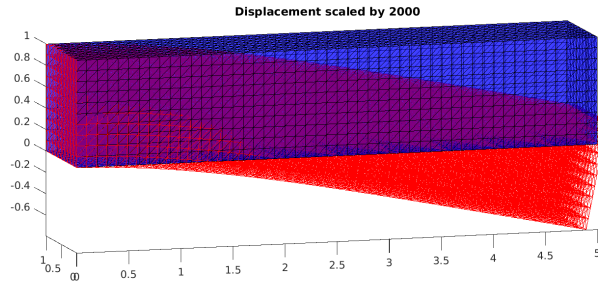


Figure 4.4: Result for the 3D elasticity problem

N	n_q		n_{me}		n_{dof}		Assembly	Solve
10	6	171	30	000	18	513	0.668 (s)	0.648 (s)
15	19	456	101	250	58	368	2.140 (s)	3.830 (s)
20	44	541	240	000	133	623	5.349 (s)	16.619 (s)
25	85	176	468	750	255	528	10.261 (s)	67.494 (s)
30	145	111	810	000	435	333	18.295 (s)	154.980 (s)

Table 4.2: Computational times for assembling and solving the 3D elasticity BVP, described in *Vector BVP 5*, with various size meshes.

remark 4.3

In the `fc_vfemp1` Matlab toolbox some codes were provided to solve the *Vector BVP 4*:

- the main script solving the BVP, and with graphic representations:
`fc_vfemp1.examples.elasticity.BVPElasticity3D01`
- the function for building the BVP with the mesh file:
`[bvp,info]=fc_vfemp1.examples.elasticity.setBVPElasticity3D01(N,verbose)`
- the commands to run the benchmarks are:

```
setBVP=@(N,verbose) fc_vfemp1.examples.elasticity.setBVPElasticity3D01(N,verbose);
fc_vfemp1.examples.bench('LN',10:5:30,'setBVP',setBVP);
```

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 4.5).

The 2D problem to solve is the following

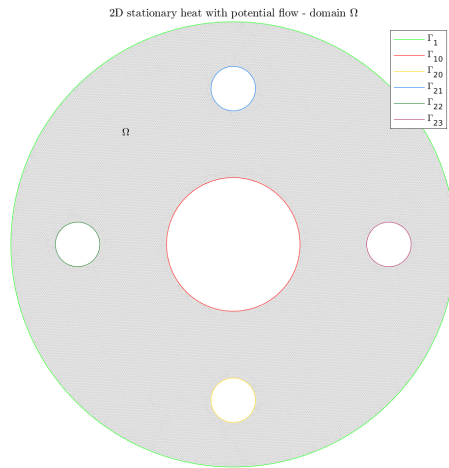


Figure 4.5: Domain and boundaries

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

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

$$-\text{div}(\alpha \nabla u) + \langle \mathbf{V}, \nabla u \rangle + \beta u = 0 \text{ in } \Omega \subset \mathbb{R}^2, \tag{4.16}$$

$$u = 20 * \mathbf{x}_2 \text{ on } \Gamma_{21}, \tag{4.17}$$

$$u = 0 \text{ on } \Gamma_{22} \cup \Gamma_{23}, \tag{4.18}$$

$$\frac{\partial u}{\partial n} = 0 \text{ on } \Gamma_1 \cup \Gamma_{10} \cup \Gamma_{20} \tag{4.19}$$

where Ω and its boundaries are given in Figure 4.5. This problem is well posed if $\alpha(\mathbf{x}) > 0$ and $\beta(\mathbf{x}) \geq 0$. We choose α and β in Ω as :

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

The potential flow is the velocity field $\mathbf{V} = \nabla \phi$ where the scalar function ϕ is the velocity potential solution of the 2D BVP (4.20)-(4.23)

Usual BVP 7 : 2D velocity potential BVP

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

$$-\Delta \phi = 0 \text{ in } \Omega, \tag{4.20}$$

$$\phi = -20 \text{ on } \Gamma_{21}, \tag{4.21}$$

$$\phi = 20 \text{ on } \Gamma_{20}, \tag{4.22}$$

$$\frac{\partial \phi}{\partial n} = 0 \text{ on } \Gamma_1 \cup \Gamma_{23} \cup \Gamma_{22} \tag{4.23}$$

Then the potential flow \mathbf{V} is solution of (4.24)

Usual vector BVP 3 : 2D potential flow

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

$$\mathbf{V} = \nabla \phi \text{ in } \Omega, \tag{4.24}$$

For a given mesh, the numerical results of the velocity potential ϕ , the potential flow \mathbf{V} and the heat u are respectively represented in Figure 4.6, 4.7 and 4.8.

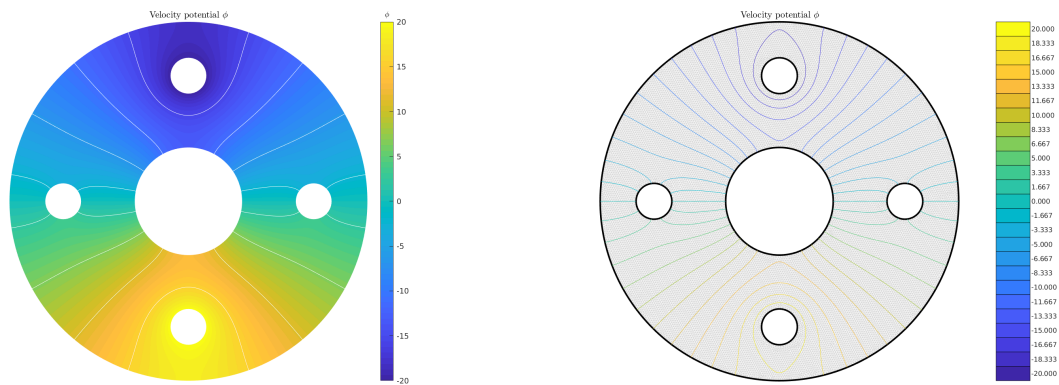


Figure 4.6: Velocity potential Φ

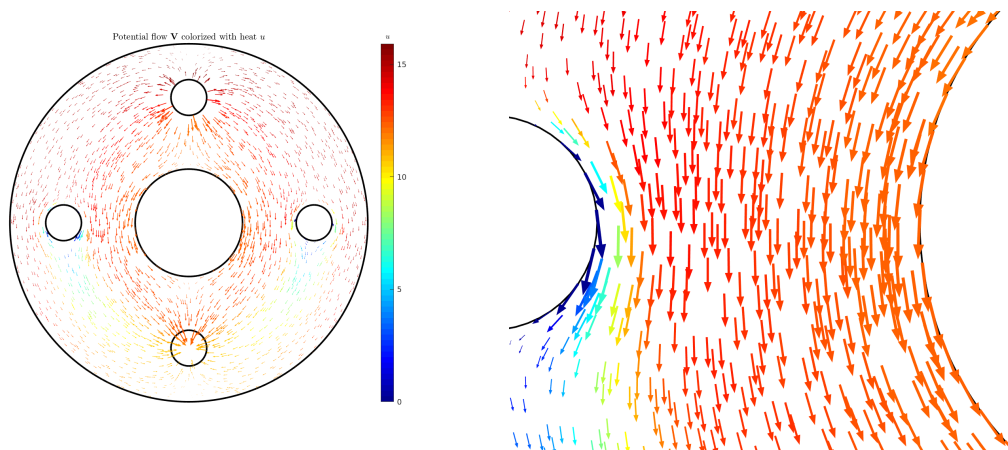


Figure 4.7: Potential flow $\mathbf{V} = \nabla \phi$ colored with u heat values

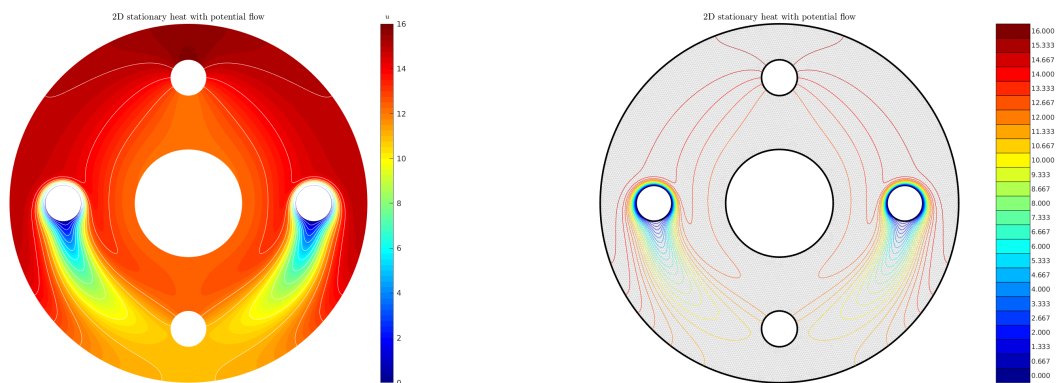



Figure 4.8: Heat u with potential flow \mathbf{V}

Now we will present two ways to solve these problems using the toolbox.

4.2.1 Method 1 : split in three parts

The 2D potential velocity problem (4.20)-(4.23) can be equivalently expressed as the scalar BVP (1.2)-(1.4) :

 **Scalar BVP 10 : 2D potential velocity**

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

$$\begin{aligned} \mathcal{L}(\phi) &= 0 && \text{in } \Omega, \\ \phi &= g^D && \text{on } \Gamma^D, \\ \frac{\partial \phi}{\partial n_{\mathcal{L}}} &= 0 && \text{on } \Gamma^R. \end{aligned}$$

where

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

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

- $\Gamma^D = \Gamma_{20} \cup \Gamma_{21}$ with $g^D := 20$ on Γ_{20} , and $g^D := -20$ on Γ_{21}
- $\Gamma^R = \Gamma_1 \cup \Gamma_{23} \cup \Gamma_{22}$

The code using the toolbox to solve (4.20)-(4.23) is given in Listing 4.7.

Listing 4.3: Stationary heat with potential flow in 2D, Matlab code (method 1)

```
geofile=fc_vfemp1.get_geo(2,2,'disk5holes.geo');
meshfile=fc_oogmsh.gmsh.buildmesh(geofile,N,'d',d);
Lop=fc_vfemp1.Loperator(d,d,{1,[],[]},1,[],[],[]);
bvpPotential=fc_vfemp1.BVP(Th,fc_vfemp1.PDE(Lop));
bvpPotential.setDirichlet(20,20);
bvpPotential.setDirichlet(21,-20);
[phi,SolveInfo]=bvpPotential.solve('time',true);
```

Now to compute \mathbf{V} , we can write the potential flow problem (4.24) with \mathcal{H} -operators as

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

where

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


The code using the toolbox for solving this problem is given in Listing 4.7.

Listing 4.4: Stationary heat with potential flow in 2D, Matlab code (method 1)

```
Hop=fc_vfemp1.Hoperator(Th.dim,d,d);
Hop.H{1,1}=fc_vfemp1.Loperator(d,d,[],[],{1,0},[]);
Hop.H{2,2}=fc_vfemp1.Loperator(d,d,[],[],{0,1},[]);
V=Hop.apply(Th,{phi,phi});
```

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

Finally, the stationary heat BVP (4.16)-(4.19) can be equivalently expressed as the scalar BVP (1.2)-(1.4) :

 **Scalar BVP 11 : 2D stationary heat**

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


$$\begin{aligned} \mathcal{L}(u) &= 0 && \text{in } \Omega, \\ u &= g^D && \text{on } \Gamma^D, \\ \frac{\partial u}{\partial n_{\mathcal{L}}} &= 0 && \text{on } \Gamma^R. \end{aligned}$$

where

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

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

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

The code using the toolbox  to solve *Scalar* BVP 11 is given in Listing 4.7.

Listing 4.5: Stationary heat with potential flow in 2D, Matlabcode (method 1)

```
Lop=fc_vfemP1.Loperator(d,d,{af,[],[]},af,[],V,b);
bvpHeat=fc_vfemP1.BVP(Th,fc_vfemP1.PDE(Lop));
bvpHeat.setDirichlet(21,gD);
bvpHeat.setDirichlet(22,0);
bvpHeat.setDirichlet(23,0);
[u,SolveInfo2]=bvpHeat.solve('time',true);
```

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

We can merged velocity potential BVP (4.20)-(4.23) and potential flow (4.24) to obtain the new BVP



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

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

$$-\left(\frac{\partial \mathbf{V}_1}{\partial x} + \frac{\partial \mathbf{V}_2}{\partial y}\right) = 0 \text{ in } \Omega, \quad (4.25)$$

$$\mathbf{V}_1 - \frac{\partial \phi}{\partial x} = 0 \text{ in } \Omega, \quad (4.26)$$

$$\mathbf{V}_2 - \frac{\partial \phi}{\partial y} = 0 \text{ in } \Omega, \quad (4.27)$$

$$\phi = -20 \text{ on } \Gamma_{21}, \quad (4.28)$$

$$\phi = 20 \text{ on } \Gamma_{20}, \quad (4.29)$$

$$\frac{\partial \phi}{\partial n} = 0 \text{ on } \Gamma_1 \cup \Gamma_{23} \cup \Gamma_{22} \quad (4.30)$$

We can also replace (4.25) by $-\Delta \phi = 0$.

Let $\mathbf{w} = \begin{pmatrix} \phi \\ \mathbf{V}_1 \\ \mathbf{V}_2 \end{pmatrix}$, the previous problem (4.25)-(4.30) can be equivalently expressed as the vector BVP (1.18)-(1.20) :



Vector BVP 6 : Velocity potential and potential flow in 2D

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

$$\mathcal{H}(\mathbf{w}) = \mathbf{0} \quad \text{in } \Omega, \quad (4.31)$$

$$\mathbf{w}_\alpha = g_\alpha^D \quad \text{on } \Gamma_\alpha^D, \quad \forall \alpha \in \llbracket 1, 3 \rrbracket, \quad (4.32)$$

$$\frac{\partial \mathbf{w}}{\partial n_{\mathcal{H}_\alpha}} = g_\alpha^R \quad \text{on } \Gamma_\alpha^R, \quad \forall \alpha \in \llbracket 1, 3 \rrbracket, \quad (4.33)$$

where

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

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

- $\Gamma_\alpha^R = \Gamma_\alpha^D = \emptyset$ for all $\alpha \in \{2, 3\}$ (no boundary conditions on \mathbf{V}_1 and \mathbf{V}_2)
- $\Gamma_1^D = \Gamma_{20} \cup \Gamma_{21}$ 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}

Indeed, to compute the conormal derivatives of \mathcal{H} we remark that

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

So we obtain

$$\frac{\partial \mathbf{w}}{\partial n_{\mathcal{H}_1}} \stackrel{\text{def}}{=} \sum_{\alpha=1}^3 \frac{\partial \mathbf{w}_\alpha}{\partial n_{\mathcal{H}_{1,\alpha}}} = \langle \mathbf{V}, \mathbf{n} \rangle = \frac{\partial \phi}{\partial \mathbf{n}}, \quad (4.34)$$

and

$$\frac{\partial \mathbf{w}}{\partial n_{\mathcal{H}_2}} = \frac{\partial \mathbf{w}}{\partial n_{\mathcal{H}_3}} := 0. \quad (4.35)$$

From (4.35), we cannot impose boundary conditions on components 2 and 3.

By using the toolbox fc_vfemp1 the Matlab code which solves the *Vector BVP 6* is very short and it is given in Listing 4.6.

Listing 4.6: Stationary heat with potential flow in 2D, Matlab code (method 1)

```
d=2;
Hop=fc_vfemp1.Hoperator(d,d,3);
Hop.set(1,2,fc_vfemp1.Loperator(d,d,[],{1,0},[],[]));
Hop.set(1,3,fc_vfemp1.Loperator(d,d,[],{0,1},[],[]));
Hop.set(2,1,fc_vfemp1.Loperator(d,d,[],{1,0},[],[]));
Hop.set(2,2,fc_vfemp1.Loperator(d,d,[],[],{1,1}));
Hop.set(3,1,fc_vfemp1.Loperator(d,d,[],[],{0,1},[],[]));
Hop.set(3,3,fc_vfemp1.Loperator(d,d,[],[],{1,1}));
bvpFlow=fc_vfemp1.BVP(Th,fc_vfemp1.PDE(Hop));
bvpFlow.setDirichlet(20,20,1);
bvpFlow.setDirichlet(21,20,1);
[W,SolveInfo]=bvpFlow.solve('split',true,'time',true);
```

Thereafter the code using the toolbox fc_vfemp1 for solving (4.16)-(4.19) is given in Listing 4.6

Listing 4.7: Stationary heat with potential flow in 2D, Matlab code (method 2)

```
Lop=fc_vfemp1.Loperator(d,d,{af,[],[];af,[],[]},{W{2},W{3}},b);
bvpHeat=fc_vfemp1.BVP(Th,fc_vfemp1.PDE(Lop));
bvpHeat.setDirichlet(21,gD);
bvpHeat.setDirichlet(22,0);
bvpHeat.setDirichlet(23,0);
[u,SolveInfo2]=bvpHeat.solve('time',true);
```

4.3 Stationary heat with potential flow in 3D

Let $\Omega \subset \mathbb{R}^3$ be the cylinder given in Figure 4.9.

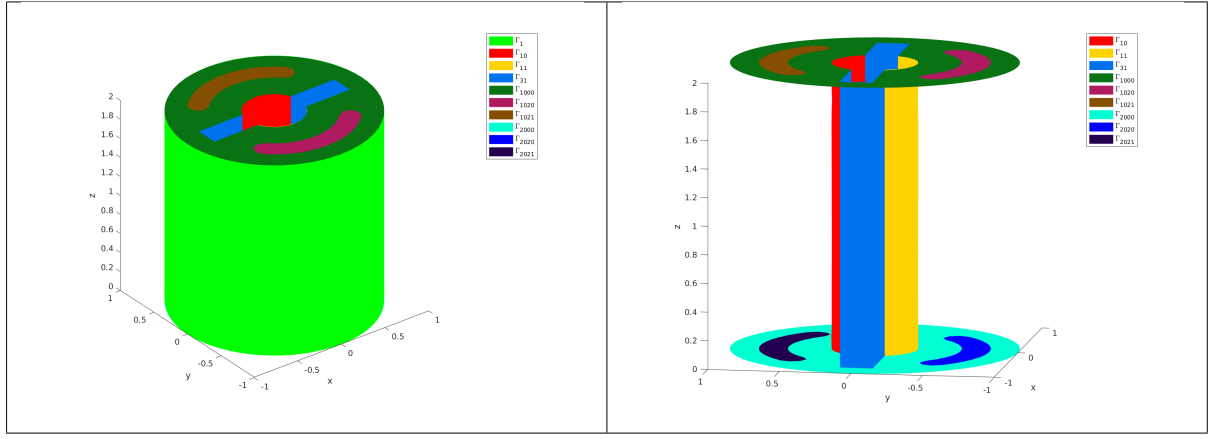


Figure 4.9: Stationary heat with potential flow : 3d mesh

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

The 3D problem to solve is the following

💡 Usual BVP 8 : 3D stationary heat with potential flow

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

$$-\operatorname{div}(\alpha \nabla u) + \langle \mathbf{V}, \nabla u \rangle + \beta u = 0 \text{ in } \Omega \subset \mathbb{R}^3, \quad (4.36)$$

$$u = 30 \text{ on } \Gamma_{1020} \cup \Gamma_{2020}, \quad (4.37)$$

$$u = 10\delta_{|z-1|>0.5} \text{ on } \Gamma_{10}, \quad (4.38)$$

$$\frac{\partial u}{\partial n} = 0 \text{ otherwise} \quad (4.39)$$

where Ω and its boundaries are given in Figure 4.9. This problem is well posed if $\alpha(\mathbf{x}) > 0$ and $\beta(\mathbf{x}) \geq 0$. We choose α and β in Ω as :

$$\alpha(\mathbf{x}) = 1 + (x_3 - 1)^2, ,$$

$$\beta(\mathbf{x}) = 0.01$$

The potential flow is the velocity field $\mathbf{V} = \nabla \phi$ where the scalar function ϕ is the velocity potential solution of the 3D BVP (4.40)-(4.43)

💡 Usual BVP 9 : 3D velocity potential

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

$$-\Delta \phi = 0 \text{ in } \Omega, \quad (4.40)$$

$$\phi = 1 \text{ on } \Gamma_{1021} \cup \Gamma_{2021}, \quad (4.41)$$

$$\phi = -1 \text{ on } \Gamma_{1020} \cup \Gamma_{2020}, \quad (4.42)$$

$$\frac{\partial \phi}{\partial n} = 0 \text{ otherwise} \quad (4.43)$$

Then the potential flow \mathbf{V} is solution of (4.44)

💡 Usual vector BVP 5 : 3D potential flow

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

$$\mathbf{V} = \nabla \phi \text{ in } \Omega, \quad (4.44)$$

For a given mesh, the numerical results of the velocity potential ϕ , the potential flow \mathbf{V} and the heat u are respectively represented in Figure 4.10, 4.11 and 4.12.

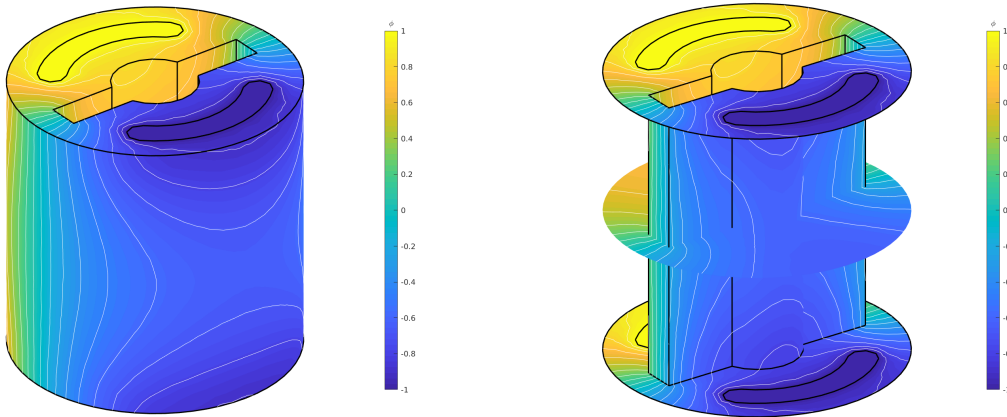


Figure 4.10: Velocity potential ϕ

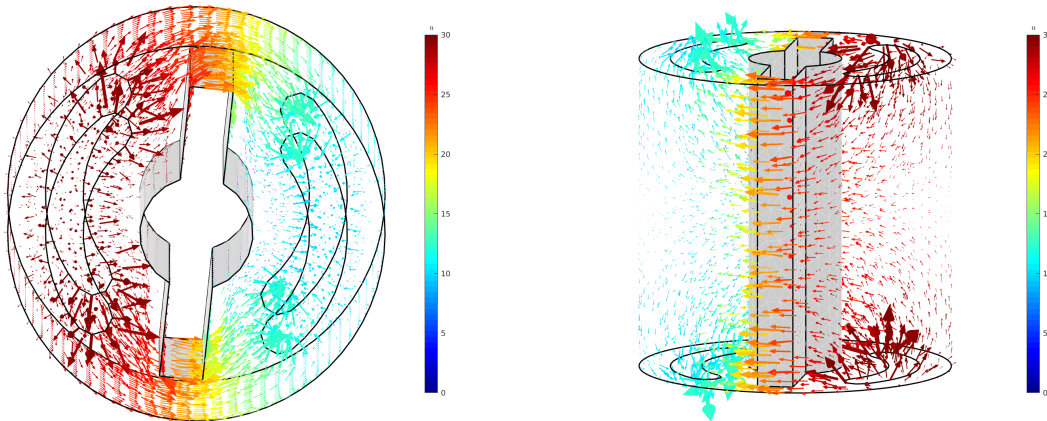


Figure 4.11: Potential flow $\mathbf{V} = \nabla \phi$ colored with u heat values

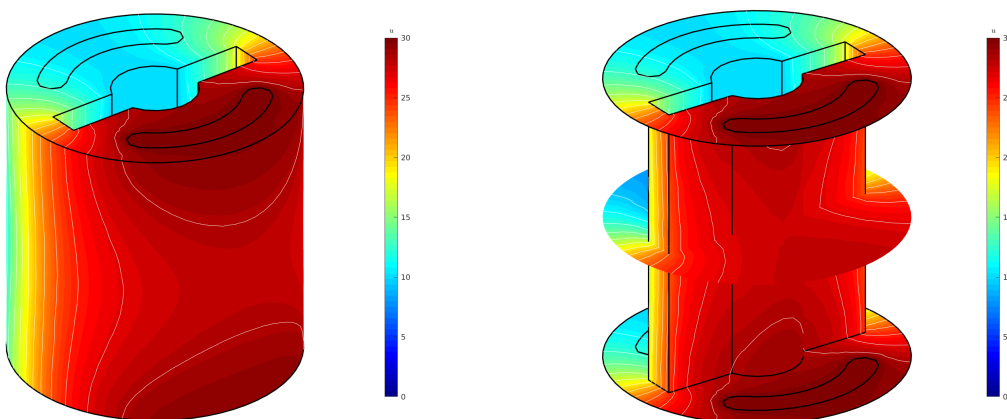


Figure 4.12: Heat solution u

Now we will present two ways to solve these problems using the [@vfmP1](#) Matlab toolbox.

4.3.1 Method 1 : split in three parts

The 3D potential velocity problem (4.40)-(4.43) can be equivalently expressed as the scalar BVP (1.2)-(1.4) :

Scalar BVP 12 : 3D potential velocity

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

$$\begin{aligned} \mathcal{L}(\phi) &= 0 && \text{in } \Omega, \\ \phi &= g^D && \text{on } \Gamma^D, \\ \frac{\partial \phi}{\partial n_{\mathcal{L}}} &= 0 && \text{on } \Gamma^R. \end{aligned}$$

where

- $\mathcal{L} := \mathcal{L}_{1,0,0,0}$, and then the conormal derivative of ϕ is given by

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

- $\Gamma^D = \Gamma_{1020} \cup \Gamma_{1021} \cup \Gamma_{2020} \cup \Gamma_{2021}$
- $g^D := 1$ on $\Gamma_{1021} \cup \Gamma_{2021}$, and $g^D := -1$ on $\Gamma_{1020} \cup \Gamma_{2020}$
- $\Gamma^R = \Gamma_1 \cup \Gamma_{10} \cup \Gamma_{11} \cup \Gamma_{31} \cup \Gamma_{1000} \cup \Gamma_{2000}$

The code solving the *Scalar BVP 12* by using the  toolbox is given in Listing 4.8. All codes presented here are parts of the script `fc_vfemp1.examples.HeatAndFlowVelocity.BVPHeatAndFlowVelocity3D01`.

Listing 4.8: Stationary heat with potential flow in 3D, Matlab code (method 1)

```
N=10;dim=3;d=3;
geofile=fc_vfemp1.get_geo(dim,d,'cylinderkey.geo');
meshfile=fc_oogmsh.gmsh.buildmesh(geofile,N,'d',d);
Th=fc_simesh.siMesh(meshfile,'dim',3);
Lop=fc_vfemp1.Loperator(dim,d,{1,[],[],1,[],[],1},[],[],[]);
bvpFlow=fc_vfemp1.BVP(Th,fc_vfemp1.PDE(Lop));
bvpFlow.setDirichlet(1021,1.);
bvpFlow.setDirichlet(2021,1.);
bvpFlow.setDirichlet(1020,1.);
bvpFlow.setDirichlet(2020,1.);
[Phi,SolveInfo]=bvpFlow.solve('time',true);
```

Now to compute \mathbf{V} , we can write the potential flow problem (4.44)

- with \mathcal{H} -operators as


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

where

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

- with \mathcal{L} -operators as


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

The code using the  toolbox for solving this problem with \mathcal{L} -operators is given in Listing 4.9.

Listing 4.9: Stationary heat with potential flow in 3D, Matlab code (method 1)

```
V=cell(1,3);
Lop=fc_vfemp1.Loperator(dim,d,[],[],{1,0,0},[]);
V{1}=Lop.apply(Th,Phi);
Lop=fc_vfemp1.Loperator(dim,d,[],[],{0,1,0},[]);
V{2}=Lop.apply(Th,Phi);
Lop=fc_vfemp1.Loperator(dim,d,[],[],{0,0,1},[]);
V{3}=Lop.apply(Th,Phi);
```

Finally, the stationary heat BVP (4.36)-(??) can be equivalently expressed as the scalar BVP (1.2)-(1.4) :

 **Scalar BVP 13 : 3D stationary heat**

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


$$\begin{aligned} \mathcal{L}(u) &= 0 && \text{in } \Omega, \\ u &= g^D && \text{on } \Gamma^D, \\ \frac{\partial u}{\partial n_{\mathcal{L}}} &= 0 && \text{on } \Gamma^R. \end{aligned}$$

where

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

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

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

The code solving the *Scalar BVP 13* by using the  toolbox is given in Listing 4.10.

Listing 4.10: Stationary heat with potential flow in 3D, Matlab code (method 1)

```
af=@(x,y,z) 1+(z 1).^2;
gD10=@(x,y,z) 10*(abs(z 1)>0.5);
b=0.01;
Lop=fc_vfemp1.Loperator(dim,d,{af,[],[],[];af,[],[],[];af,[],[],[]},{V{1},V{2},V{3}},b);
bvpHeat=fc_vfemp1.BVP(Th,fc_vfemp1.PDE(Lop));
bvpHeat.setDirichlet(1020,30.);
bvpHeat.setDirichlet(2020,30.);
bvpHeat.setDirichlet(10,gD10);
[u,SolveInfo2]=bvpHeat.solve('time',true);
```

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

To solve problem (4.36)-(4.39), we need to compute the velocity field \mathbf{V} . For that we can rewrite the potential flow problem (4.40)-(4.43), by introducing $\mathbf{V} = (\mathbf{V}_1, \mathbf{V}_2, \mathbf{V}_3)$ as unknowns :



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

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

$$-\left(\frac{\partial \mathbf{V}_1}{\partial x} + \frac{\partial \mathbf{V}_2}{\partial y} + \frac{\partial \mathbf{V}_3}{\partial z}\right) = 0 \text{ in } \Omega, \quad (4.45)$$

$$\mathbf{V}_1 - \frac{\partial \phi}{\partial x} = 0 \text{ in } \Omega, \quad (4.46)$$

$$\mathbf{V}_2 - \frac{\partial \phi}{\partial y} = 0 \text{ in } \Omega, \quad (4.47)$$

$$\mathbf{V}_3 - \frac{\partial \phi}{\partial z} = 0 \text{ in } \Omega, \quad (4.48)$$

with boundary conditions (4.41) to (4.43).

We can also replace (4.45) by $-\Delta\phi = 0$.

Let $\mathbf{w} = \begin{pmatrix} \phi \\ \mathbf{V}_1 \\ \mathbf{V}_2 \\ \mathbf{V}_3 \end{pmatrix}$, the previous PDE can be written as a vector boundary value problem (see section 1.3) where the \mathcal{H} -operator is given by

$$\mathcal{H}(\mathbf{w}) = 0 \quad (4.49)$$

with

$$\mathcal{H}_{1,1} = 0, \quad \mathcal{H}_{1,2} = \mathcal{L}_{0,-\mathbf{e}_1,0,0}, \quad \mathcal{H}_{1,3} = \mathcal{L}_{0,-\mathbf{e}_2,0,0}, \quad \mathcal{H}_{1,4} = \mathcal{L}_{0,-\mathbf{e}_3,0,0}, \quad (4.50)$$

$$\mathcal{H}_{2,1} = \mathcal{L}_{0,0,-\mathbf{e}_1,0}, \quad \mathcal{H}_{2,2} = \mathcal{L}_{0,0,0,1}, \quad \mathcal{H}_{2,3} = 0, \quad \mathcal{H}_{2,4} = 0, \quad (4.51)$$

$$\mathcal{H}_{3,1} = \mathcal{L}_{0,0,-\mathbf{e}_2,0}, \quad \mathcal{H}_{3,2} = 0, \quad \mathcal{H}_{3,3} = \mathcal{L}_{0,0,0,1}, \quad \mathcal{H}_{3,4} = 0, \quad (4.52)$$

$$\mathcal{H}_{4,1} = \mathcal{L}_{0,0,-\mathbf{e}_3,0}, \quad \mathcal{H}_{4,2} = 0, \quad \mathcal{H}_{4,3} = 0, \quad \mathcal{H}_{4,4} = \mathcal{L}_{0,0,0,1}, \quad (4.53)$$

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

The conormal derivatives are given by

$$\begin{array}{cccc} \frac{\partial \mathbf{w}_1}{\partial n_{\mathcal{H}_{1,1}}} = 0, & \frac{\partial \mathbf{w}_1}{\partial n_{\mathcal{H}_{2,1}}} = 0, & \frac{\partial \mathbf{w}_1}{\partial n_{\mathcal{H}_{3,1}}} = 0, & \frac{\partial \mathbf{w}_1}{\partial n_{\mathcal{H}_{4,1}}} = 0, \\ \frac{\partial \mathbf{w}_2}{\partial n_{\mathcal{H}_{1,2}}} = \mathbf{V}_1 \mathbf{n}_1, & \frac{\partial \mathbf{w}_2}{\partial n_{\mathcal{H}_{2,2}}} = 0, & \frac{\partial \mathbf{w}_2}{\partial n_{\mathcal{H}_{3,2}}} = 0, & \frac{\partial \mathbf{w}_2}{\partial n_{\mathcal{H}_{4,2}}} = 0, \\ \frac{\partial \mathbf{w}_3}{\partial n_{\mathcal{H}_{1,3}}} = \mathbf{V}_2 \mathbf{n}_2, & \frac{\partial \mathbf{w}_3}{\partial n_{\mathcal{H}_{2,3}}} = 0, & \frac{\partial \mathbf{w}_3}{\partial n_{\mathcal{H}_{3,3}}} = 0, & \frac{\partial \mathbf{w}_3}{\partial n_{\mathcal{H}_{4,3}}} = 0, \\ \frac{\partial \mathbf{w}_4}{\partial n_{\mathcal{H}_{1,4}}} = \mathbf{V}_3 \mathbf{n}_3, & \frac{\partial \mathbf{w}_4}{\partial n_{\mathcal{H}_{2,4}}} = 0, & \frac{\partial \mathbf{w}_4}{\partial n_{\mathcal{H}_{3,4}}} = 0, & \frac{\partial \mathbf{w}_4}{\partial n_{\mathcal{H}_{4,4}}} = 0, \end{array}$$

So we obtain

$$\sum_{\alpha=1}^4 \frac{\partial \mathbf{w}_\alpha}{\partial n_{\mathcal{H}_{1,\alpha}}} = \langle \mathbf{V}, \mathbf{n} \rangle = \langle \nabla \phi, \mathbf{n} \rangle, \quad (4.54)$$

and

$$\sum_{\alpha=1}^4 \frac{\partial \mathbf{w}_\alpha}{\partial n_{\mathcal{H}_{2,\alpha}}} = \sum_{\alpha=1}^4 \frac{\partial \mathbf{w}_\alpha}{\partial n_{\mathcal{H}_{3,\alpha}}} = \sum_{\alpha=1}^4 \frac{\partial \mathbf{w}_\alpha}{\partial n_{\mathcal{H}_{4,\alpha}}} = 0. \quad (4.55)$$

From (4.55), we cannot impose boundary conditions on components 2 to 4. Thus, with notation of section 1.3, 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 (4.41) to (4.43), we set $\Gamma_1^D = \Gamma_{1020} \cup \Gamma_{1021} \cup \Gamma_{2020} \cup \Gamma_{2021}$, $\Gamma_1^N = \Gamma \setminus \Gamma_1^D$ and $g_1^D = \delta_{\Gamma_{1020} \cup \Gamma_{2020}} - \delta_{\Gamma_{1021} \cup \Gamma_{2021}}$, $g_1^N = 0$.

The full problem is resumed in *Vector BVP 7*.

Vector BVP 7 : Velocity potential and potential flow in 3D

Find $\mathbf{w} = (\mathbf{w}_1, \mathbf{w}_2, \mathbf{w}_3, \mathbf{w}_4) \in (\mathbb{H}^2(\Omega))^4$ such that

$$\mathcal{H}(\mathbf{w}) = \mathbf{0} \quad \text{in } \Omega, \quad (4.56)$$

$$\mathbf{w}_\alpha = g_\alpha^D \quad \text{on } \Gamma_\alpha^D, \quad \forall \alpha \in \llbracket 1, 4 \rrbracket, \quad (4.57)$$

$$\frac{\partial \mathbf{w}}{\partial n_{\mathcal{H}_\alpha}} = 0 \quad \text{on } \Gamma_\alpha^R, \quad \forall \alpha \in \llbracket 1, 4 \rrbracket, \quad (4.58)$$

where

- \mathcal{H} is the 4-by-4 operator defined by

$$\mathcal{H} = \begin{pmatrix} 0 & \mathcal{L}_{\mathbf{0}, -\mathbf{e}_1, \mathbf{0}, \mathbf{0}} & \mathcal{L}_{\mathbf{0}, -\mathbf{e}_2, \mathbf{0}, \mathbf{0}} & \mathcal{L}_{\mathbf{0}, -\mathbf{e}_3, \mathbf{0}, \mathbf{0}} \\ \mathcal{L}_{\mathbf{0}, \mathbf{0}, -\mathbf{e}_1, \mathbf{0}} & \mathcal{L}_{\mathbf{0}, \mathbf{0}, \mathbf{0}, 1} & 0 & 0 \\ \mathcal{L}_{\mathbf{0}, \mathbf{0}, -\mathbf{e}_2, \mathbf{0}} & 0 & \mathcal{L}_{\mathbf{0}, \mathbf{0}, \mathbf{0}, 1} & 0 \\ \mathcal{L}_{\mathbf{0}, \mathbf{0}, -\mathbf{e}_3, \mathbf{0}} & 0 & 0 & \mathcal{L}_{\mathbf{0}, \mathbf{0}, \mathbf{0}, 1} \end{pmatrix}$$

- $\Gamma_\alpha^R = \Gamma_\alpha^D = \emptyset$ for all $\alpha \in \{2, 3, 4\}$ (no boundary conditions on V_1, V_2 and V_3)
- $\Gamma_1^D = \Gamma_{1020} \cup \Gamma_{1021} \cup \Gamma_{2020} \cup \Gamma_{2021}$ and $\Gamma_1^R = \Gamma \setminus \Gamma_1^D$
- $g_1^D = \delta_{\Gamma_{1020} \cup \Gamma_{2020}} - \delta_{\Gamma_{1021} \cup \Gamma_{2021}}$,

The code using the toolbox for solving (4.45)-(4.48) is given in Listing 4.11

Listing 4.11: Stationary heat with potential flow in 3D, Matlab code (method 2)

```
dim=3;d=3;m=4;N=10;
geofile=fc_vfemp1.get_geo(dim,d,'cylinderkey.geo');
meshfile=fc_oogmsh.gmsh.buildmesh(geofile,N,'d',d);
Th=fc_simesh.siMesh(meshfile);
Hop=fc_vfemp1.Hoperator(dim,d,m);
Hop.set(1,2,fc_vfemp1.Loperator(dim,d,[],{1,0,0},[],[]));
Hop.set(1,3,fc_vfemp1.Loperator(dim,d,[],{0,1,0},[],[]));
Hop.set(1,4,fc_vfemp1.Loperator(dim,d,[],{0,0,1},[],[]));
Hop.set(2,1,fc_vfemp1.Loperator(dim,d,[],[],{1,0,0},[]));
Hop.set(2,2,fc_vfemp1.Loperator(dim,d,[],[],[],1));
Hop.set(3,1,fc_vfemp1.Loperator(dim,d,[],[],{0,1,0},[]));
Hop.set(3,3,fc_vfemp1.Loperator(dim,d,[],[],[],1));
Hop.set(4,1,fc_vfemp1.Loperator(dim,d,[],[],{0,0,1},[]));
Hop.set(4,4,fc_vfemp1.Loperator(dim,d,[],[],[],1));
bvpPotentials=fc_vfemp1.BVP(Th,fc_vfemp1.PDE(Hop));
bvpPotentials.setDirichlet(1020,1,1);
bvpPotentials.setDirichlet(1021,1,1);
bvpPotentials.setDirichlet(2020,1,1);
bvpPotentials.setDirichlet(2021,1,1);
[W,SolveInfo]=bvpPotentials.solve('split',true,'time',true);
```

Thereafter, to compute the heat u we solve the *Scalar* BVP 13 described in previous section. The code is given in Listing 4.12

Listing 4.12: Stationary heat with potential flow in 3D, Matlab code (method 2)

```
af=@(x,y,z) 1+(z 1).^2;
gD10=@(x,y,z) 10*(abs(z 1)>0.5);
b=0.01;
Lop=fc_vfemp1.Loperator(dim,d,{af,[],[],[],af,[],[],af,[],{W{2},W{3},W{4}},b);
bvpHeat=fc_vfemp1.BVP(Th,fc_vfemp1.PDE(Lop));
bvpHeat.setDirichlet(1020,30.);
bvpHeat.setDirichlet(2020,30.);
bvpHeat.setDirichlet(10,gD10);
[u,SolveInfo2]=bvpHeat.solve('time',true);
```


Chapter 5

Other problems

With the  Matlab toolbox add-on tools are provided. They allow to easily distribute codes using the toolbox. Actually, some add-ons are in development to solve particular boundary value problems:

- **electrostatic** BVP,
- **biharmonic** BVP,
- **eigenvalues** BVP,
- **surface** BVP,
- ...

Appendices

1 Linear elasticity

1.1 Elasticity in \mathbb{R}^d

Mathematical notations

We want to prove (4.5) of Lemma 4.1. We can write (4.1) as

$$-(\operatorname{div} \sigma(\mathbf{u}))_i = f_i, \quad \forall i \in \llbracket 1, d \rrbracket, \text{ in } \Omega \quad (1)$$

We have

$$\begin{aligned} (\operatorname{div} \sigma(\mathbf{u}))_i &= (2\mu\epsilon_{ij}(\mathbf{u}))_{,j} + (\lambda\epsilon_{kk}(\mathbf{u}))_{,i} \\ &= \sum_{j=1}^d \frac{\partial}{\partial x_j} (2\mu\epsilon_{ij}(\mathbf{u})) + \frac{\partial}{\partial x_i} \left(\lambda \sum_{k=1}^d \epsilon_{kk}(\mathbf{u}) \right) \end{aligned}$$

and

$$\epsilon_{ij}(\mathbf{u}) = \frac{1}{2} \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right)$$

So we obtain

$$\begin{aligned} (\operatorname{div} \sigma(\mathbf{u}))_i &= \sum_{j=1}^d \frac{\partial}{\partial x_j} \left(\mu \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i} \right) \right) + \frac{\partial}{\partial x_i} \left(\lambda \sum_{j=1}^d \frac{\partial u_j}{\partial x_j} \right) \\ &= \sum_{j=1}^d \left\{ \frac{\partial}{\partial x_j} \left(\mu \frac{\partial u_i}{\partial x_j} \right) + \frac{\partial}{\partial x_j} \left(\mu \frac{\partial u_j}{\partial x_i} \right) \right\} + \sum_{j=1}^d \frac{\partial}{\partial x_i} \left(\lambda \frac{\partial u_j}{\partial x_j} \right) \\ &= \sum_{k=1}^d \frac{\partial}{\partial x_k} \left(\mu \frac{\partial u_i}{\partial x_k} \right) + \sum_{j=1}^d \frac{\partial}{\partial x_j} \left(\mu \frac{\partial u_j}{\partial x_i} \right) + \sum_{j=1}^d \frac{\partial}{\partial x_i} \left(\lambda \frac{\partial u_j}{\partial x_j} \right) \\ &= \sum_{j=1}^d \left\{ \sum_{k=1}^d \frac{\partial}{\partial x_k} \left(\mu \frac{\partial u_j}{\partial x_k} \right) \delta_{ij} + \frac{\partial}{\partial x_j} \left(\mu \frac{\partial u_j}{\partial x_i} \right) \right\} + \sum_{j=1}^d \frac{\partial}{\partial x_i} \left(\lambda \frac{\partial u_j}{\partial x_j} \right) \end{aligned}$$

So, from (1.18) and (1) we want $\forall i \in \llbracket 1, d \rrbracket$

$$(\mathcal{H}(\mathbf{u}))_i = \sum_{j=1}^d \mathcal{H}_{i,j}(\mathbf{u}_j) = (\operatorname{div} \sigma(\mathbf{u}))_i.$$

and by identification, we obtain $\forall i \in \llbracket 1, d \rrbracket$

$$\mathcal{H}_{i,j}(\mathbf{u}_j) = \frac{\partial}{\partial x_j} \left(\mu \frac{\partial u_j}{\partial x_i} \right) + \frac{\partial}{\partial x_i} \left(\lambda \frac{\partial u_j}{\partial x_j} \right) + \sum_{k=1}^d \frac{\partial}{\partial x_k} \left(\mu \frac{\partial u_j}{\partial x_k} \right) \delta_{ij}$$

and then $\mathcal{H}_{i,j} := \mathcal{L}_{\mathbb{A}^{i,j}, \mathbf{0}, \mathbf{0}, \mathbf{0}}$ with

$$(\mathbb{A}^{i,j})_{k,l} = \mu \delta_{k,l} \delta_{i,j} + \mu \delta_{k,j} \delta_{l,i} + \lambda \delta_{k,i} \delta_{l,j}, \quad \forall (k,l) \in \llbracket 1, d \rrbracket^2,$$

With these notations, we can rewrite the elasticity problem (1) as

$$\sum_{j=1}^d \operatorname{div}(\mathbb{A}^{i,j} \nabla \mathbf{u}_j) + f_i = 0, \quad \forall i \in \llbracket 1, d \rrbracket, \text{ in } \Omega \quad (2)$$

In dimension $d = 2$, (2) becomes

$$\operatorname{div} \left(\begin{pmatrix} \gamma & 0 \\ 0 & \mu \end{pmatrix} \nabla \mathbf{u}_1 \right) + \operatorname{div} \left(\begin{pmatrix} 0 & \lambda \\ \mu & 0 \end{pmatrix} \nabla \mathbf{u}_2 \right) + f_1 = 0, \quad (3)$$

$$\operatorname{div} \left(\begin{pmatrix} 0 & \mu \\ \lambda & 0 \end{pmatrix} \nabla \mathbf{u}_1 \right) + \operatorname{div} \left(\begin{pmatrix} \mu & 0 \\ 0 & \gamma \end{pmatrix} \nabla \mathbf{u}_2 \right) + f_2 = 0, \quad (4)$$

and in dimension $d = 3$

$$\operatorname{div} \left(\begin{pmatrix} \gamma & 0 & 0 \\ 0 & \mu & 0 \\ 0 & 0 & \mu \end{pmatrix} \nabla \mathbf{u}_1 \right) + \operatorname{div} \left(\begin{pmatrix} 0 & \lambda & 0 \\ \mu & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \nabla \mathbf{u}_2 \right) + \operatorname{div} \left(\begin{pmatrix} 0 & 0 & \lambda \\ 0 & 0 & 0 \\ \mu & 0 & 0 \end{pmatrix} \nabla \mathbf{u}_3 \right) + f_1 = 0, \quad (5)$$

$$\operatorname{div} \left(\begin{pmatrix} 0 & \mu & 0 \\ \lambda & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \nabla \mathbf{u}_1 \right) + \operatorname{div} \left(\begin{pmatrix} \mu & 0 & 0 \\ 0 & \gamma & 0 \\ 0 & 0 & \mu \end{pmatrix} \nabla \mathbf{u}_2 \right) + \operatorname{div} \left(\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & \lambda \\ 0 & \mu & 0 \end{pmatrix} \nabla \mathbf{u}_3 \right) + f_2 = 0, \quad (6)$$

$$\operatorname{div} \left(\begin{pmatrix} 0 & 0 & \mu \\ 0 & 0 & 0 \\ \lambda & 0 & 0 \end{pmatrix} \nabla \mathbf{u}_1 \right) + \operatorname{div} \left(\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & \mu \\ 0 & \lambda & 0 \end{pmatrix} \nabla \mathbf{u}_2 \right) + \operatorname{div} \left(\begin{pmatrix} \mu & 0 & 0 \\ 0 & \mu & 0 \\ 0 & 0 & \gamma \end{pmatrix} \nabla \mathbf{u}_3 \right) + f_3 = 0, \quad (7)$$

Boundary conditions

We want to prove (4.6) of Lemma 4.1. We set $\frac{\partial \mathbf{u}_j}{\partial n_{i,j}} := \frac{\partial \mathbf{u}_j}{\partial n_{i,j}}$ and, by definition of $\mathcal{H}_{i,j}$ operators, we obtain on Γ

$$\sum_{j=1}^d \frac{\partial \mathbf{u}_j}{\partial n_{i,j}} = \langle \mathbb{A}^{i,i} \nabla \mathbf{u}_i, \mathbf{n} \rangle + \sum_{\substack{j=1 \\ j \neq i}}^d \langle \mathbb{A}^{i,j} \nabla \mathbf{u}_j, \mathbf{n} \rangle$$

But we have

$$\begin{aligned} \langle \mathbb{A}^{i,i} \nabla \mathbf{u}_i, \mathbf{n} \rangle &= \sum_{k=1}^d \sum_{l=1}^d \mathbb{A}_{k,l}^{i,i} \frac{\partial \mathbf{u}_j}{\partial x_l} \mathbf{n}_k \\ &= \sum_{k=1}^d \sum_{l=1}^d (\mu \delta_{k,l} + (\lambda + \mu) \delta_{k,i} \delta_{l,i}) \frac{\partial \mathbf{u}_i}{\partial x_l} \mathbf{n}_k \\ &= \mu \sum_{k=1}^d \frac{\partial \mathbf{u}_i}{\partial x_k} \mathbf{n}_k + (\lambda + \mu) \frac{\partial \mathbf{u}_i}{\partial x_i} \end{aligned}$$

and, for $j \neq i$

$$\begin{aligned} \langle \mathbb{A}^{i,j} \nabla \mathbf{u}_j, \mathbf{n} \rangle &= \sum_{k=1}^d \sum_{l=1}^d \mathbb{A}_{k,l}^{i,j} \frac{\partial \mathbf{u}_j}{\partial x_l} \mathbf{n}_k \\ &= \sum_{k=1}^d \sum_{l=1}^d (\lambda \delta_{k,i} \delta_{l,j} + \mu \delta_{k,j} \delta_{l,i}) \frac{\partial \mathbf{u}_j}{\partial x_l} \mathbf{n}_k \\ &= \lambda \frac{\partial \mathbf{u}_j}{\partial x_j} \mathbf{n}_i + \mu \frac{\partial \mathbf{u}_j}{\partial x_i} \mathbf{n}_j. \end{aligned}$$

So we obtain

$$\sum_{j=1}^d \frac{\partial \mathbf{u}_j}{\partial n_{i,j}} = \sum_{\substack{j=1 \\ j \neq i}}^d \left(\lambda \frac{\partial \mathbf{u}_j}{\partial x_j} \mathbf{n}_i + \mu \frac{\partial \mathbf{u}_j}{\partial x_i} \mathbf{n}_j \right) + \mu \sum_{k=1}^d \frac{\partial \mathbf{u}_i}{\partial x_k} \mathbf{n}_k + (\lambda + \mu) \frac{\partial \mathbf{u}_i}{\partial x_i} \mathbf{n}_i \quad (8)$$

In linear elasticity boundary conditions can be expressed as *Dirichlet* or as $(\sigma(\vec{u})\mathbf{n})_i = g$ for example. We have

$$(\sigma(\vec{u})\mathbf{n})_i = \sum_{j=1}^d \sigma_{i,j}(\mathbf{u}) \mathbf{n}_j$$

with

$$\sigma_{i,j}(\mathbf{u}) = 2\mu \epsilon_{i,j}(\mathbf{u}) + \lambda \delta_{i,j} \sum_{k=1}^d \epsilon_{k,k}(\mathbf{u}) \quad \text{and} \quad \epsilon_{i,j}(\mathbf{u}) = \frac{1}{2} \left(\frac{\partial \mathbf{u}_j}{\partial x_i} + \frac{\partial \mathbf{u}_i}{\partial x_j} \right)$$

So we have with $\sigma_{i,j}(\mathbf{u}) = 2\mu\epsilon_{i,j}(\mathbf{u}) + \lambda\delta_{i,j} \sum_{k=1}^d \epsilon_{k,k}(\mathbf{u})$

$$\begin{aligned} (\sigma(\vec{u})\mathbf{n})_i &= \sigma_{i,i}(\mathbf{u})\mathbf{n}_i + \sum_{\substack{j=1 \\ j \neq i}}^d \sigma_{i,j}(\mathbf{u})\mathbf{n}_j \\ &= \left(2\mu\epsilon_{i,i}(\mathbf{u}) + \lambda \sum_{k=1}^d \epsilon_{k,k}(\mathbf{u}) \right) \mathbf{n}_i + \sum_{\substack{j=1 \\ j \neq i}}^d 2\mu\epsilon_{i,j}(\mathbf{u})\mathbf{n}_j \end{aligned}$$

We also have $\epsilon_{i,j}(\mathbf{u}) = \frac{1}{2} \left(\frac{\partial \mathbf{u}_j}{\partial x_i} + \frac{\partial \mathbf{u}_i}{\partial x_j} \right)$ and then

$$\begin{aligned} (\sigma(\vec{u})\mathbf{n})_i &= \sum_{\substack{j=1 \\ j \neq i}}^d \mu \left(\frac{\partial \mathbf{u}_j}{\partial x_i} + \frac{\partial \mathbf{u}_i}{\partial x_j} \right) \mathbf{n}_j + \left(2\mu \frac{\partial \mathbf{u}_i}{\partial x_i} + \lambda \sum_{k=1}^d \frac{\partial \mathbf{u}_k}{\partial x_k} \right) \mathbf{n}_i \\ &= \sum_{\substack{j=1 \\ j \neq i}}^d \mu \frac{\partial \mathbf{u}_j}{\partial x_i} \mathbf{n}_j + \sum_{\substack{j=1 \\ j \neq i}}^d \mu \frac{\partial \mathbf{u}_i}{\partial x_j} \mathbf{n}_j + \lambda \sum_{\substack{k=1 \\ k \neq i}}^d \frac{\partial \mathbf{u}_k}{\partial x_k} \mathbf{n}_i + (\lambda + 2\mu) \frac{\partial \mathbf{u}_i}{\partial x_i} \mathbf{n}_i \\ &= \sum_{\substack{j=1 \\ j \neq i}}^d \left(\mu \frac{\partial \mathbf{u}_j}{\partial x_i} \mathbf{n}_j + \lambda \frac{\partial \mathbf{u}_j}{\partial x_j} \mathbf{n}_i \right) + \sum_{j=1}^d \mu \frac{\partial \mathbf{u}_i}{\partial x_j} \mathbf{n}_j + (\lambda + \mu) \frac{\partial \mathbf{u}_i}{\partial x_i} \mathbf{n}_i. \end{aligned}$$

So we have proved that

$$(\sigma(\vec{u})\mathbf{n})_i = \sum_{j=1}^d \frac{\partial \mathbf{u}_j}{\partial n_{i,j}}, \quad \forall i \in \llbracket 1, d \rrbracket \quad (9)$$

Listings

1.1	Complete Matlab code to solve the 2D condenser problem with graphical representations	9
1.2	Complete Matlab code to solve the simple 2D vector BVP with graphical representations	11
2.1	2D example : apply method	16
3.1	Poisson 2D BVP with Dirichlet boundary conditions : numerical solution (left) and error (right)	25
3.2	Poisson 2D BVP with mixed boundary conditions : numerical solution (left) and error (right)	28
3.3	3D Poisson BVP with mixed boundary conditions : numerical solution (upper) and error (bottom)	32
3.4	4D Poisson BVP with mixed boundary conditions	35
3.5	1D BVP with mixed boundary conditions	37
3.6	Setting the 2D stationary convection-diffusion BVP and representation of the numerical solution. Part of the file +fc_vfemp1/+examples/setBVPStationaryConvectionDiffusion2D01.m	39
	benchs/BVPStationaryConvectionDiffusion2D01_Matlab2019a.m	40
3.7	Setting the 3D stationary convection-diffusion BVP and representations of the numerical solution. Part of the file +fc_vfemp1/+examples/setBVPStationaryConvectionDiffusion3D01.m	42
	benchs/BVPStationaryConvectionDiffusion3D01_Matlab2019a.m	42
3.8	Setting the 2D electrostatic BVP, Matlab code	44
	benchs/BVPElectrostatic2D01_Matlab2019a.m	46
4.1	2D elasticity, Matlab code	49
	benchs/BVPElasticity2D01_Matlab2019a.m	50
4.2	3D elasticity, Matlab code	51
	benchs/BVPElasticity3D01_Matlab2019a.m	52
4.3	Stationary heat with potential flow in 2D, Matlab code (method 1)	55
4.4	Stationary heat with potential flow in 2D, Matlab code (method 1)	55
4.5	Stationary heat with potential flow in 2D, Matlabcode (method 1)	56
4.6	Stationary heat with potential flow in 2D, Matlab code (method 1)	57
4.7	Stationary heat with potential flow in 2D, Matlab code (method 2)	57
4.8	Stationary heat with potential flow in 3D, Matlab code (method 1)	60

4.9	Stationary heat with potential flow in 3D, Matlab code (method 1)	61
4.10	Stationary heat with potential flow in 3D, Matlab code (method 1)	61
4.11	Stationary heat with potential flow in 3D, Matlab code (method 2)	63
4.12	Stationary heat with potential flow in 3D, Matlab code (method 2)	63

Bibliography

- [1] F. Cuvelier. `fc_oogmsh`: an object-oriented Matlab toolbox to run **gmsh** and read mesh files. <http://www.math.univ-paris13.fr/~cuvelier/software/>, 2017. User's Guide.
- [2] F. Cuvelier. `fc_simesh`: an object-oriented Matlab toolbox for using simplices meshes generated from gmsh (in dimension 2 or 3) or an hypercube triangulation (in any dimension). <http://www.math.univ-paris13.fr/~cuvelier/software/>, 2017. User's Guide.
- [3] F. Cuvelier and G. Scarella. A generic way to solve partial differential equations by the \mathbb{P}_1 -Lagrange finite element method in vector languages. https://www.math.univ-paris13.fr/~cuvelier/software/docs/Recherch/VecFEM/distrib/0.1b1/vecFEMP1_report-0.1b1.pdf, 2015.
- [4] François Cuvelier, Caroline Japhet, and Gilles Scarella. An efficient way to assemble finite element matrices in vector languages. *BIT Numerical Mathematics*, 56(3):833–864, dec 2015.
- [5] G. Dhatt, E. Lefrançois, and G. Touzot. *Finite Element Method*. Wiley, 2012.
- [6] A. Quarteroni and A. Valli. *Numerical Approximation of Partial Differential Equations*, volume 23 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, 1994.

Informations for git maintainers of the Matlab toolbox

git informations on the toolboxes used to build this manual

```

-----
name : fc-vfemP1
tag : 0.2.0
commit : ca743a7c582948b6db8f00c987b41d40bf4d6c5b
date : 2020-03-06
time : 07-38-29
status : 0
-----
name : fc-tools
tag : 0.0.31
commit : 3a25f08630ff69e7f8ef4ab451671e454c492e36
date : 2020-02-23
time : 11-27-44
status : 0
-----
name : fc-bench
tag : 0.1.2
commit : 666dc60d1277f5fa9c99dee4ae1c33270f22c57d
date : 2020-02-16
time : 06-38-46
status : 0
-----
name : fc-hypermesh
tag : 1.0.3
commit : c520b34cfd7eb0dbf9e4ecd459fd7162db73cc58
date : 2020-02-16
time : 08-34-19
status : 0
-----
name : fc-amat
tag : 0.1.1
commit : ab639dd1d3be0ed76a8df9a9b637400ad797f32a
date : 2020-01-02
time : 06-40-28
status : 0
-----
name : fc-meshtools
tag : 0.1.3
commit : cdbc41bc98af4e4faccc1746024aced1f21aae53
date : 2020-02-17
time : 10-52-56
status : 0
-----
name : fc-graphics4mesh
tag : 0.1.2
commit : 84c21b1d58ce72ea0b5bbafa2f23c03ab5b61bbf
date : 2020-02-23
time : 14-41-38
status : 0
-----
name : fc-oogmsh
tag : 0.2.3
commit : 614c8154b265a17a6dd31e328100974ea7e9e0bc
date : 2020-03-06
time : 05-19-37
status : 0
-----
name : fc-siplt
tag : 0.2.1
commit : 5e2479703ae79cb78ab966c34508bfcfd1bbfa9
date : 2020-02-23
time : 14-43-43
status : 0
-----
name : fc-simesh
tag : 0.4.1
commit : a3da1f107b97164de3f7076ee312fa7696903468
date : 2020-03-03
time : 07-58-04
status : 0
-----

```

git informations on the L^AT_EX package used to build this manual

```

-----
name : fctools
tag :
commit : 57968c4a96c2593ccc9da9efd3e52b2ff012cb5
date : 2020-02-07
time : 06:41:09
status : 1
-----

```

Using the remote configuration repository:

```

url      ssh://lagagit/MCS/Cuvelier/Matlab/fc-config
commit  0cc9a09a69da15bc4bd78a80e8f05a4c7e0456db

```