fc vfem$\mathbb{P}_1$ Octave package, User's Guide[1]

version 0.2.1

François Cuvelier[2]

Thursday 19th March, 2020

[1] LaTeX manual, revision 0.2.1, compiled with Octave 5.2.0, and packages `fc-vfemp1`[0.2.1], `fc-tools`[0.0.31], `fc-bench`[0.1.2], `fc-hypermesh`[1.0.3], `fc-amat`[0.1.2], `fc-meshtools`[0.1.3], `fc-graphics4mesh`[0.1.3], `fc-oogmsh`[0.2.3], `fc-siplt`[0.2.2], `fc-simesh`[0.4.2]

[2] Université 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.

vfem$\mathbb{P}_1$ is an object-oriented Octave package 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 package 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 package is used.

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

Contents

Chapter 1

Presentation

Firstly, the installation process of the fc vfemP1 package is presented. Thereafter a generic scalar boundary value problem is given by using notations of [?]. A simple example of such problem is solved with this package. 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 (recommanded)

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

ofc_vfemp1_install.m

or get it on the dedicated web page. Thereafter, one run it under Octave. This command download, extract and configure the *fc-vfemp1* and the required packages (*fc-tools*, *fc-oogmsh*, *fc-hypermesh*) in the current directory.

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

```
>> cd ~/Octave/packages
>> mfc_vfemp1_install
```

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

```
Parts of the GNU Octave <fc-vfemp1> package.
Copyright (C) 2017 Francois Cuvelier <cuvelier@math.univ-paris13.fr>

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

1- Downloading and extracting the packages
   -> <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 packages
2-a) Setting the <fc-hypermesh> package
Write in ~/Octave/packages/fc-vfemp1-full/fc_hypermesh-0.0.6/configure_loc.m ...
  -> done
2-b) Setting the <fc-oogmsh> package
[fc-oogmsh] Using GMSH binary : ~/bin/gmsh
[fc-oogmsh] Writing in ~/Octave/packages/fc-vfemp1-full/fc_oogmsh-0.0.17/configure_loc.m ...
[fc-oogmsh] configured with
   ->     gmsh_bin='~/bin/gmsh';
   ->     mesh_dir='~/Octave/packages/fc-vfemp1-full/fc_oogmsh-0.0.17/meshes';
   ->      geo_dir='~/Octave/packages/fc-vfemp1-full/fc_oogmsh-0.0.17/geodir';
   -> fc_tools_dir='~/Octave/packages/./fc-vfemp1-full/fc_tools-0.0.19';
[fc-oogmsh] done
2-c) Setting the <fc-simesh> package without graphics
[fc-simesh] Unable to load the fc-siplt toolbox/package in current path
[fc-simesh] Guess path does not exists:
   -> siplt
[fc-] Guess path does not exists:
   -> [fc-simesh] Use fc_simesh.configure('fc_siplt_dir',<DIR>) to correct this issue

[fc-simesh] no graphics package installed
[fc-simesh] Writing in ~/Octave/packages/fc-vfemp1-full/fc_simesh-0.2.1/configure_loc.m ...
[fc-simesh] configured with
   -> oogmsh_dir    ='~/Octave/packages/./fc-vfemp1-full/fc_oogmsh-0.0.17';
   -> hypermesh_dir ='~/Octave/packages/./fc-vfemp1-full/fc_hypermesh-0.0.6';
   -> siplt_dir     ='';
[fc-simesh] done
2-d) Setting the <fc-graphics4mesh> toolbox
Write in ~/Octave/packages/fc-vfemp1-full/fc_graphics4mesh-0.0.2/configure_loc.m ...
  -> done
2-e) Setting the <fc-siplt> toolbox
Write in ~/Octave/packages/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 ~/Octave/packages/fc-vfemp1-full/fc_simesh-0.2.1/configure_loc.m ...
[fc-simesh] configured with
   -> oogmsh_dir    ='~/Octave/packages/./fc-vfemp1-full/fc_oogmsh-0.0.17';
   -> hypermesh_dir ='~/Octave/packages/./fc-vfemp1-full/fc_hypermesh-0.0.6';
   -> siplt_dir     ='~/Octave/packages/./fc-vfemp1-full/fc_siplt-0.0.2';
[fc-simesh] done
3- Using instructions
   To use the <fc-simesh> package:
   addpath('~/Octave/packages/./fc-vfemp1-full/fc_simesh-0.2.1')
   fc_simesh.init()

   See ~/Octave/packages/ofc_simesh_set.m
    <fc-simesh>[0.2.1]: installed

**************************************
Downloading and installing the package
    <fc-vfemp1>[0.1.0]
**************************************
*** Setting the <fc-vfemp1> package
Write in ~/Octave/packages/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> package:
   addpath('~/Octave/packages/./fc-vfemp1-full/fc_vfemp1-0.1.0')
   fc_vfemp1.init()

   See ~/Octave/packages/ofc_vfemp1_set.m
```

The complete package (i.e. with all the other needed packages) is stored in the directory `~/Octave/packages/fc-vfe` and, for each Octave session, one have to set the package by:

```
>> addpath('~/Octave/packages/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* package without graphical extension one can use the following command

```
>> ofc_vfemp1_install('graphics',false)
```

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

`~/Octave/packages/fc-vfemp1-full`

1.1.2 Other kind

to do!

1.2 Scalar boundary value problem

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

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

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

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

A *scalar* boundary value problem is given by

Scalar BVP 1 : generic problem

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

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

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

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

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

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

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

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

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

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

Scalar VF 1 : generic problem

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

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

where

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

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

To have an outline of the fc-vfemP1 package, 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 $u \in \mathrm{H}^2(\Omega)$ such that

$$-\Delta u \;=\; 0 \;\text{ in } \Omega \subset \mathbb{R}^2, \tag{1.10}$$

$$u \;=\; 0 \;\text{ on } \Gamma_1, \tag{1.11}$$

$$u \;=\; -12 \;\text{ on } \Gamma_{98}, \tag{1.12}$$

$$u \;=\; 12 \;\text{ on } \Gamma_{99}, \tag{1.13}$$

where Ω and its boundaries are given in Figure 1.1.

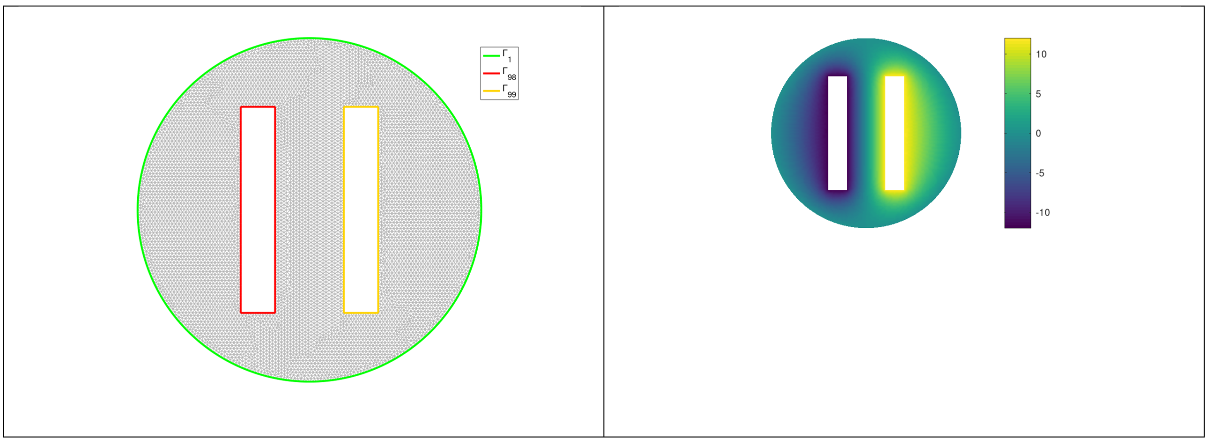


Figure 1.1: 2D condenser mesh and boundaries (left) and numerical solution (right)

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

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

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

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

In Listing 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 Octave 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	$\mathrm{n_q}$	$\mathrm{n_{me}}$	Assembly	Solve
100	10 201	20 000	0.086 (s)	0.052 (s)
200	40 401	80 000	0.137 (s)	0.187 (s)
300	90 601	180 000	0.225 (s)	0.478 (s)
400	160 801	320 000	0.366 (s)	0.954 (s)
500	251 001	500 000	0.537 (s)	1.626 (s)
600	361 201	720 000	0.793 (s)	2.416 (s)
700	491 401	980 000	1.298 (s)	3.461 (s)
800	641 601	1 280 000	1.727 (s)	4.572 (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

$$\left\{ \begin{array}{cccc} \mathcal{H} \; : & \left(\mathrm{H}^2(\Omega)\right)^m & \longrightarrow & \left(L^2(\Omega)\right)^m \\ & \boldsymbol{u} = (\boldsymbol{u}_1, \ldots, \boldsymbol{u}_m) & \longmapsto & \boldsymbol{f} = (\boldsymbol{f}_1, \ldots, \boldsymbol{f}_m) \stackrel{\text{def}}{=} \mathcal{H}(\boldsymbol{u}) \end{array} \right. \tag{1.14}$$

where

$$\boldsymbol{f}_\alpha = \sum_{\beta=1}^{m} \mathcal{H}_{\alpha,\beta}(\boldsymbol{u}_\beta), \quad \forall \alpha \in [\![1, m]\!], \tag{1.15}$$

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

$$\mathcal{H}_{\alpha,\beta} \stackrel{\text{def}}{=} \mathcal{L}_{\mathbb{A}^{\alpha,\beta}, \boldsymbol{b}^{\alpha,\beta}, \boldsymbol{c}^{\alpha,\beta}, a_0^{\alpha,\beta}} \tag{1.16}$$

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

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

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

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

A *vector* boundary value problem is given by

Vector BVP 1 : generic problem

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

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

$$\boldsymbol{u}_\alpha = g_\alpha^D \quad \text{on } \Gamma_\alpha^D,\ \forall \alpha \in [\![1, m]\!], \tag{1.19}$$

$$\frac{\partial \boldsymbol{u}}{\partial n_{\mathcal{H}_\alpha}} + a_\alpha^R \boldsymbol{u}_\alpha = g_\alpha^R \quad \text{on } \Gamma_\alpha^R,\ \forall \alpha \in [\![1, m]\!], \tag{1.20}$$

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

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

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 $\frac{\partial \boldsymbol{u}}{\partial n_{\mathcal{H}_1}} + a_1^R \boldsymbol{u}_1 = g_1^R$ and a Dirichlet one with $\boldsymbol{u}_2 = g_2^D$.

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

Variational form of the *vector* BVP

Find $\boldsymbol{u} \in H^1_{g_1^D,\Gamma_1^D} \times \dots \times H^1_{g_m^D,\Gamma_m^D}$ such that

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

where

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

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

where

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

To have an outline of the fc vfem$\mathbb{P}_1$ package, 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 $\boldsymbol{u} = (u_1, u_2) \in (\mathrm{H}^2(\Omega))^2$ such that

$$\begin{aligned}
-\Delta u_1 + u_2 &= 0 \quad \text{in } \Omega \subset \mathbb{R}^2, & (1.26)\\
-\Delta u_2 + u_1 &= 0 \quad \text{in } \Omega \subset \mathbb{R}^2, & (1.27)\\
(u_1, u_2) &= (0, 0) \quad \text{on } \Gamma_1, & (1.28)\\
(u_1, u_2) &= (-12., +12.) \quad \text{on } \Gamma_{98}, & (1.29)\\
(u_1, u_2) &= (+12., -12.) \quad \text{on } \Gamma_{99}, & (1.30)
\end{aligned}$$

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

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

where

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

$$f \equiv 0,$$

and

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

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 Octave 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
```

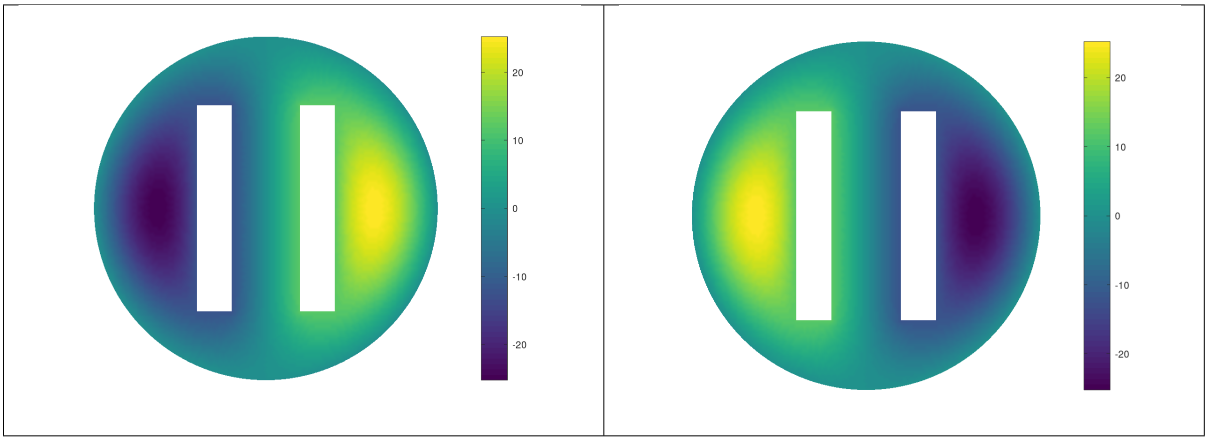


Figure 1.2: 2D simple vector BVP, u_1 numerical solution (left) and u_2 numerical solution (right)

N	$\mathrm{n_q}$	$\mathrm{n_{me}}$	$\mathrm{n_{dof}}$	Assembly	Solve
10	8 151	15 708	16 302	0.144 (s)	0.156 (s)
20	31 742	62 296	63 484	0.333 (s)	0.682 (s)
30	70 744	139 706	141 488	0.773 (s)	1.760 (s)
40	124 930	247 484	249 860	1.582 (s)	3.474 (s)
50	194 775	386 580	389 550	2.724 (s)	5.834 (s)
60	279 962	556 360	559 924	4.704 (s)	8.974 (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 $\mathcal{L}_{\mathbb{A},\boldsymbol{b},\boldsymbol{c},a_0}$ defined in (1.1). Its main properties are

Properties of fc_vfemp1.Loperator object

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

2.2.1 Constructor

Its contructor 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

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

in $\mathbb{R}$ Lop=fc_vfemp1.Loperator (1,1,{1},[],[],[])
in $\mathbb{R}^2$ Lop=fc_vfemp1.Loperator (2,2,{1,[];[],1},[],[],[])
in $\mathbb{R}^3$ Lop=fc_vfemp1.Loperator (3,3,{1,[],[];[],1,[];[],[],1},[],[],[])
⋮

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

in $\mathbb{R}$ Lop=fc_vfemp1.Loperator(1,1,{1},[],[],1)
in $\mathbb{R}^2$ Lop=fc_vfemp1.Loperator (2,2,{1,[];[],1},[],[],1)
in $\mathbb{R}^3$ Lop=fc_vfemp1.Loperator (3,3,{1,[],[];[],1,[];[],[],1},[],[],1)
⋮

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

```
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

$$\mathcal{L}^1 \stackrel{\mathsf{def}}{=} \mathcal{L}_{\mathbb{O},\mathbf{0},\boldsymbol{c},a_0}$$

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

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

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

$$\int_\Omega w.v d\mathrm{q} = \int_\Omega \langle \boldsymbol{c}, \boldsymbol{\nabla}\, u \rangle\, v d\mathrm{q} + \int_\Omega a_0 u v d\mathrm{q}, \quad \forall v \in \mathrm{L}^2(\Omega).$$

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 div(u)

In dimension 2, we have

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

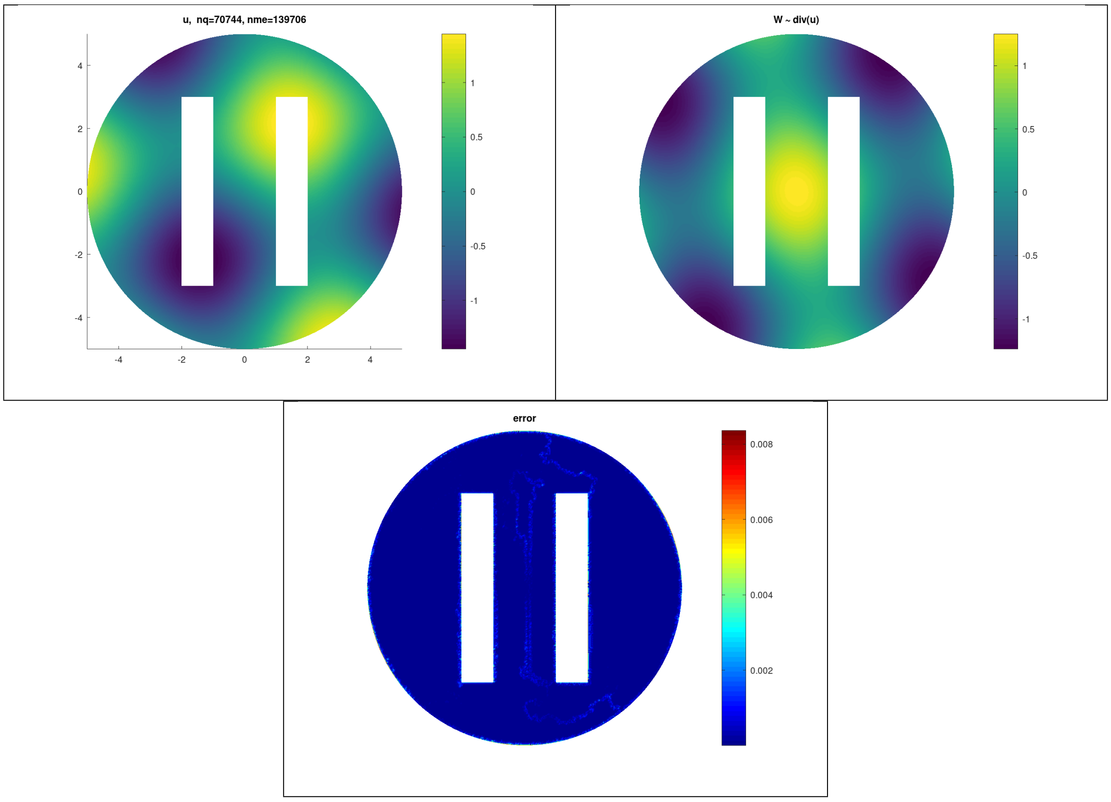
So to create this operator under Octave one just has to do

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

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

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

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




```
u=@(x,y) 3/2*cos(x/2 y/3).*sin(x/3+y/2);
divu=@(x,y) 3/2*(5/6*cos(1/2*x   1/3*y).*cos(1/3*x + 1/2*y)   1/6*sin(1/2*x   1/3*y).*sin(1/3*x + 1/2*y));

meshfile=fc_oogmsh.gmsh.buildmesh2d('condenser',30); % generate mesh
Th=fc_simesh.siMesh(meshfile); % read mesh
Lop=fc_vfemp1.Loperator(2,2,[],[],{1,1},[]);
W=Lop.apply(Th,u);

% Graphic parts
figure(1)
Th.plot(Th.eval(u))
shading interp
axis image;colorbar
title(sprintf('u,  nq=%d, nme=%d',Th.nq,Th.get_nme()))

figure(2)
Th.plot(W)
shading interp
axis off,axis image;colorbar
title('W ~ div(u)')

figure(3)
Th.plot(abs(W Th.eval(divu)))
colormap('jet')
shading interp
axis off,axis image;colorbar
title('error')
```

Listing 2.1: 2D example : apply method

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

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

2.3.1 Constructor

Its contructor 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 $\boldsymbol{u} = (u_1, u_2)$ the operator $\mathcal{H}$ defined by

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

could be written as

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

and then

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

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

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

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 contructor 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 package and and more particularly with the siMesh object.

The properties of the object BVP are

Properties of BVP object

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

2.5.1 Constructor

Its contructor 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 choosen 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

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

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

$$u_i = \mathrm{g\{i\}}, \forall i \in [\![1, \mathrm{m}]\!] \text{ on } \Gamma_{\mathrm{label}}.$$

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

$$u_{\mathrm{Lm(i)}} = \mathrm{g\{i\}}, \forall i \in [\![1, \mathbf{length}(\mathrm{Lm})]\!] \text{ on } \Gamma_{\mathrm{label}}.$$

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)

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

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

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

`obj.setRobin(label,gr,ar,Lm)` for vector B.V.P., sets Robin boundary condition (1.20) :
$\forall i \in [\![1, \mathbf{length}(\mathrm{Lm})]\!]$, let $\alpha = \mathrm{Lm(i)}$ then

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

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

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

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

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

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

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

$$-\Delta \stackrel{\mathsf{def}}{=} -\sum_{i=1}^{\mathrm{dim}} \frac{\partial^2}{\partial {x_i}^2} = \mathcal{L}_{\mathbb{I},\mathbf{0},\mathbf{0},0}. \tag{3.4}$$

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

$$\frac{\partial u}{\partial n_{\mathcal{L}}} \stackrel{\mathsf{def}}{=} \langle \mathbb{A}\,\boldsymbol{\nabla}\, u, \boldsymbol{n}\rangle - \langle \boldsymbol{b} u, \boldsymbol{n}\rangle = \frac{\partial u}{\partial n}. \tag{3.5}$$

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

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 [?] for explanation and Figure 3.1 for graphic representions) by the command

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

- the **gmsh** sotfware by using the square4.geo file (see [?] for explanation and Figure 3.1 for graphic represention) and the commands

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

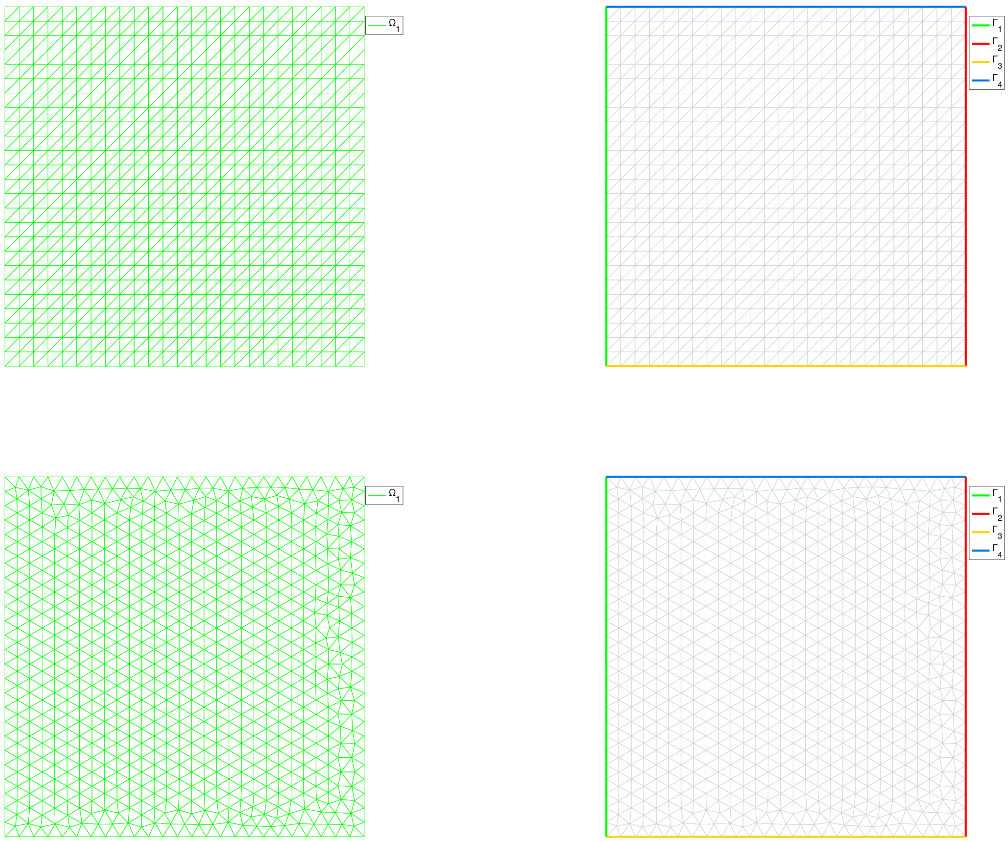


Figure 3.1: Meshes of the unit square (left) and its boundaries (right) generated with the fc_simesh.hypercube function (top) and with the **gmsh** sotfware and the square4.geo file (bottom)

We choose the problem to have the following exact solution

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

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

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

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

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

So this problem can be written as the scalar BVP 1

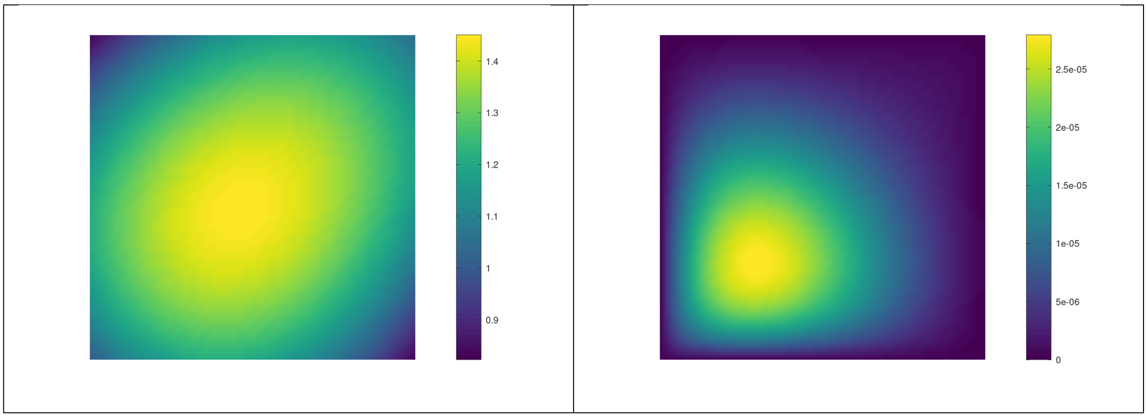
***Scalar* BVP 3 : 2D Poisson BVP with Dirichlet boundary conditions**

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

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

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

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




```
1 uex=@(x,y) cos(x y).*sin(x+y)+exp( (x.^2+y.^2));
2 f=@(x,y) 4*x.^2.*exp( x.^2 y.^2) 4*y.^2.*exp( x.^2 y.^2) + 4*cos(x y).*sin(x+y) + 4*exp( x.^2 y.^2);
3 Th=fc_simesh.hypercube(2,50); %
4 Lop=fc_vfemp1.Loperator(2,2,{1,0;0,1},[],[],[]); %
5 pde=fc_vfemp1.PDE(Lop,f); %
6 bvp=fc_vfemp1.BVP(Th,pde); %
7 for lab=1:4, bvp.setDirichlet( lab, uex);end % Setting Dirichlet boundary conditions
8 U=bvp.solve(); % Solving the BVP
```

Listing 3.1: Poisson 2D BVP with Dirichlet boundary conditions : numerical solution (left) and error (right)

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	$\mathrm{n_q}$	$\mathrm{n_{me}}$	Assembly	Solve
100	10 201	20 000	0.091 (s)	0.043 (s)
200	40 401	80 000	0.137 (s)	0.211 (s)
300	90 601	180 000	0.231 (s)	0.488 (s)
400	160 801	320 000	0.373 (s)	0.951 (s)
500	251 001	500 000	0.557 (s)	1.635 (s)
600	361 201	720 000	0.823 (s)	2.449 (s)
700	491 401	980 000	1.328 (s)	3.530 (s)
800	641 601	1 280 000	1.758 (s)	4.720 (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	0.096 (s)	0.080 (s)
200	46 681	92 560	0.184 (s)	0.377 (s)
300	104 707	208 212	0.366 (s)	0.978 (s)
400	185 703	369 804	0.702 (s)	1.941 (s)
500	290 158	578 314	1.167 (s)	3.334 (s)
600	417 242	832 082	1.898 (s)	5.043 (s)
700	567 287	1 131 772	3.148 (s)	7.358 (s)
800	741 022	1 478 842	4.214 (s)	10.148 (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	4.456e-08	2.069e-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 $\mathbb{P}^1$-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 $\mathbb{P}^1$-Lagrange finite element solution on meshes generated with the **gmsh** software and the *square4.geo* file.

remark 3.1

In the vfem$\mathbb{P}_1$ Octave package 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

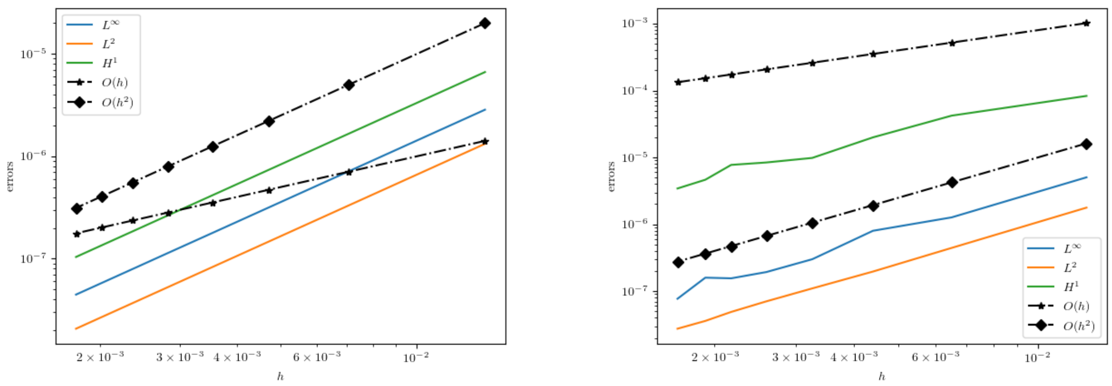


Figure 3.2: 2D Poisson BVP with dirichlet boundary conditions: order of the P_1-Lagrange finite element method in function of the mesh size h for meshes generated respectively with the fc_simesh.hypercube function (left) and with the **gmsh** software (right).

```
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

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

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

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

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

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

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

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

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

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

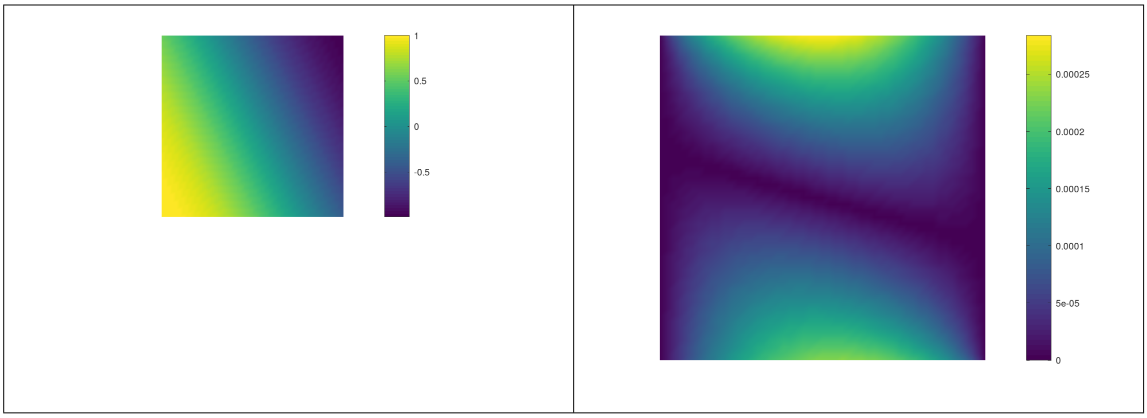
So this problem can be written as the scalar BVP 1

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

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

$$\begin{aligned}
\mathcal{L}_{\mathbb{I},\mathbf{0},\mathbf{0},0}(u) &= f \quad \text{in } \Omega = [0,1]^2, && (3.8)\\
u &= u_{\text{ex}} \quad \text{on } \Gamma_1 \cup \Gamma_2 \cup \Gamma_3 \cup \Gamma_4, && (3.9)\\
\frac{\partial u}{\partial n_{\mathcal{L}}} + a^R u &= g^R \quad \text{on } \Gamma_3, && (3.10)\\
\frac{\partial u}{\partial n_{\mathcal{L}}} &= g^N \quad \text{on } \Gamma_4, && (3.11)\\
& && (3.12)
\end{aligned}$$

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




```
1  uex=@(x,y) cos(2*x+y);
2  f=@(x,y) 5*cos(2*x+y);
3  gradu={@(x,y)  2*sin(2*x+y), @(x,y)  sin(2*x+y)};
4  ar3=@(x,y) 1+x.^2+y.^2;
5  Th=fc_simesh.hypercube(2,50);
6  Lop=fc_vfemp1.Loperator(2,2,{1,0;0,1},[],[],[]);
7  pde=fc_vfemp1.PDE(Lop,f);
8  bvp=fc_vfemp1.BVP(Th,pde);
9  bvp.setDirichlet( 1, uex);
10 bvp.setDirichlet( 2, uex);
11 bvp.setRobin( 3, @(x,y)  gradu{2}(x,y)+ar3(x,y).*uex(x,y),ar3); %
12 bvp.setRobin( 4, gradu{2},[]); %
13 U=bvp.solve();
```

Listing 3.2: Poisson 2D BVP with mixed boundary conditions : numerical solution (left) and error (right)

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	0.084 (s)	0.042 (s)
200	40 401	80 000	0.127 (s)	0.189 (s)
300	90 601	180 000	0.216 (s)	0.490 (s)
400	160 801	320 000	0.336 (s)	0.968 (s)
500	251 001	500 000	0.501 (s)	1.639 (s)
600	361 201	720 000	0.759 (s)	2.449 (s)
700	491 401	980 000	1.262 (s)	3.523 (s)
800	641 601	1 280 000	1.644 (s)	4.605 (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	0.091 (s)	0.078 (s)
200	46 681	92 560	0.175 (s)	0.377 (s)
300	104 707	208 212	0.318 (s)	0.964 (s)
400	185 703	369 804	0.593 (s)	1.857 (s)
500	290 158	578 314	0.978 (s)	3.267 (s)
600	417 242	832 082	1.688 (s)	4.894 (s)
700	567 287	1 131 772	2.560 (s)	7.172 (s)
800	741 022	1 478 842	3.613 (s)	9.929 (s)

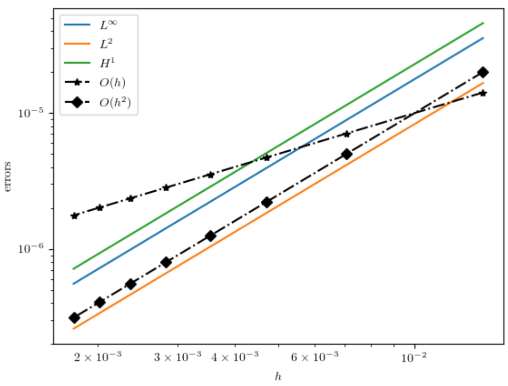
Table 3.6: Computational times for assembling and solving the 2D Poisson BVP, described in *Scalar* BVP 4, where meshes are generated with the `gmsh` software and the *square4.geo* file.

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.



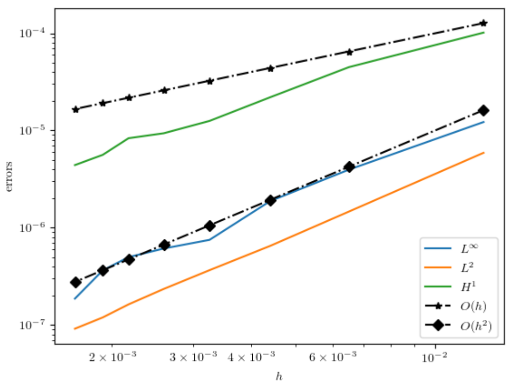


Figure 3.3: 2D Poisson BVP with mixed boundary conditions: order of the P_1-Lagrange finite element method in function of the mesh size h for meshes generated respectively with the fc_simesh.hypercube function (left) and with the `gmsh` software (right).

remark 3.2

In the fc vfemP1 Octave package 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 [?] for explanation and Figure 3.4 for graphic represention) by the command

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

- the `gmsh` sotfware by using the cube6.geo file (see [?] for explanation and Figure 3.1 for graphic represention) and the commands

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

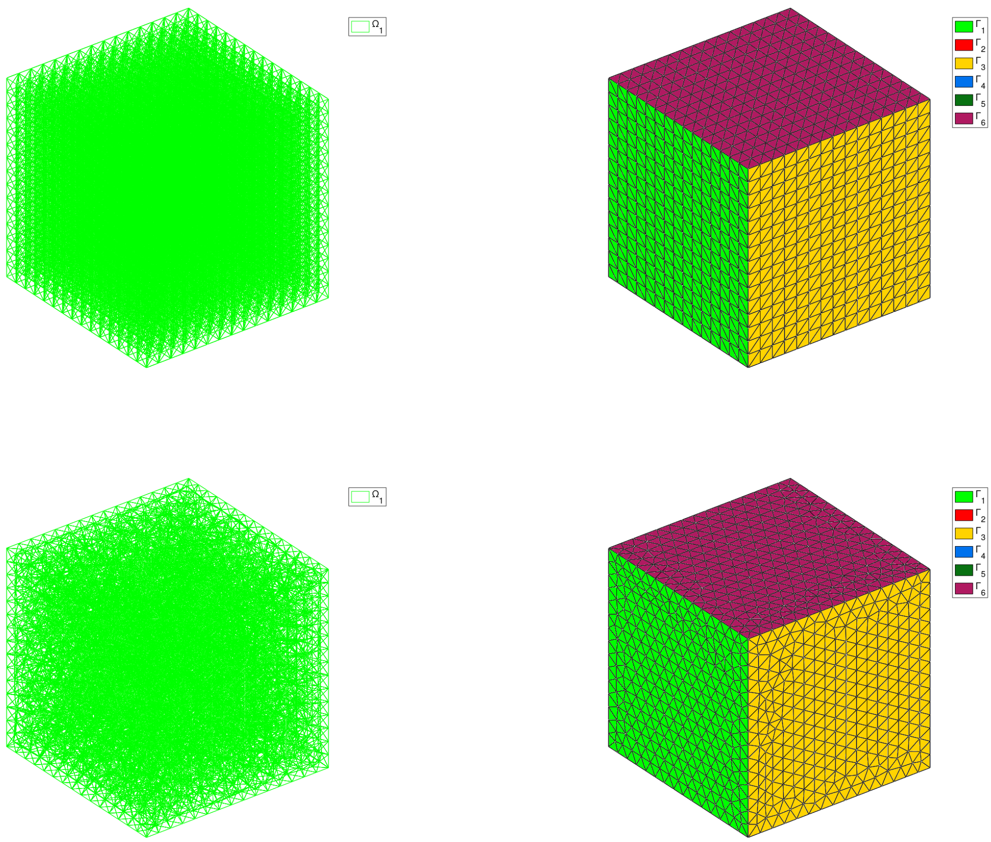


Figure 3.4: Meshes of the unit cube (left) and its boundaries (right) generated with the fc_simesh.hypercube function (top) and with the **gmsh** sotfware and the cube6.geo file (bottom)

We choose the problem to have exact solution

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

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

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

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

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

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

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

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

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

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

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

So this problem can be written as the scalar BVP 1

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

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

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

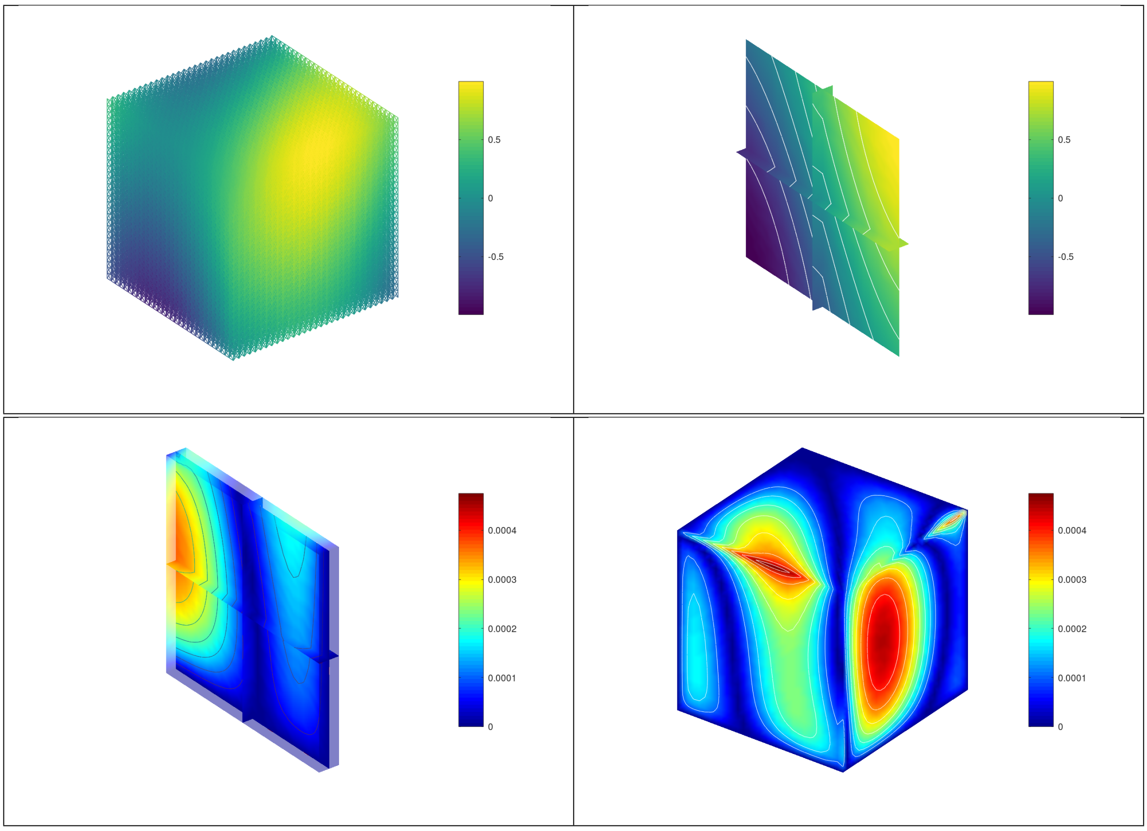
$$u = u_{\mathrm{ex}} \quad \text{on } \Gamma_1 \cup \Gamma_3 \cup \Gamma_5, \tag{3.14}$$

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

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

$$\tag{3.17}$$

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




```
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_vfemp1.Loperator(3,3,{1,0,0;0,1,0;0,0,1},[],[],[]);
pde=fc_vfemp1.PDE(Lop,f);
bvp=fc_vfemp1.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	0.187 (s)	0.383 (s)
25	17 576	93 750	0.273 (s)	1.359 (s)
30	29 791	162 000	0.386 (s)	3.653 (s)
35	46 656	257 250	0.647 (s)	6.336 (s)
40	68 921	384 000	1.029 (s)	14.559 (s)
45	97 336	546 750	1.614 (s)	32.798 (s)
50	132 651	750 000	2.233 (s)	59.205 (s)
55	175 616	998 250	2.971 (s)	62.338 (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	0.211 (s)	0.209 (s)
25	13 769	71 726	0.303 (s)	0.539 (s)
30	22 817	122 187	0.467 (s)	1.231 (s)
35	35 162	192 730	0.703 (s)	2.489 (s)
40	51 568	287 339	1.234 (s)	4.701 (s)
45	72 570	409 908	1.775 (s)	8.022 (s)
50	98 013	559 094	2.471 (s)	13.069 (s)
55	129 523	745 396	3.341 (s)	22.613 (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.

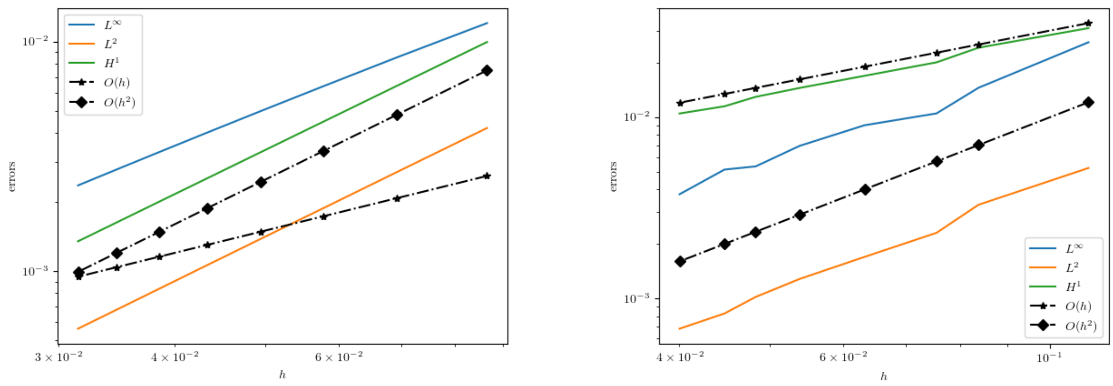


Figure 3.5: 3D Poisson BVP with mixed boundary conditions: order of the P_1-Lagrange finite element method in function of the mesh size h for meshes generated respectively with the fc_simesh.hypercube function (left) and with the `gmsh` software (right).

remark 3.3

In the fc vfem$\mathbb{P}_1$ Octave package 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 [?] for explanation by the command

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

We choose the problem to have exact solution

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

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

$$\begin{aligned} f(\boldsymbol{x}) = {} & 2\,\cos(x_1 - 2\,x_2 + x_3 - 2\,x_4)\sin(2\,x_1 - x_2 - x_3 + x_4) \\ & + 17\,\cos(2\,x_1 - x_2 - x_3 + x_4)\sin(x_1 - 2\,x_2 + x_3 - 2\,x_4)\,. \end{aligned}$$

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

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

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

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

with

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

and

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

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

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

with

$$\begin{aligned} g^N(\boldsymbol{x}) = {} & \cos(2\,x_1 - x_2 - x_3 + x_4)\cos(x_1 - 2\,x_2 + x_3 - 2\,x_4) + \sin(2\,x_1 - x_2 - x_3 + x_4)\sin(x_1 - 2\,x_2 + x_3 - 2\,x_4), \\ & \forall \boldsymbol{x} \in \Gamma_6 \end{aligned}$$

and

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

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

$$\begin{aligned} \mathcal{L}_{\mathbb{I},\mathbf{0},\mathbf{0},0}(u) &= f \ \text{ in } \Omega = [0,1]^4, && (3.18) \\ u &= u_{\mathrm{ex}} \ \text{ on } \Gamma_1 \cup \Gamma_3 \cup \Gamma_5 \cup \Gamma_7, && (3.19) \\ \frac{\partial u}{\partial n_{\mathcal{L}}} + a^R u &= g^R \ \text{ on } \Gamma_2 \cup \Gamma_4, && (3.20) \\ \frac{\partial u}{\partial n_{\mathcal{L}}} &= g^N \ \text{ on } \Gamma_6 \cup \Gamma_8, && (3.21) \\ & && (3.22) \end{aligned}$$

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

```
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)  2*sin(2*x1  x2  x3 + ...
   x4).*sin(x1  2*x2 + x3  2*x4), ...
      @(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), ...
      @(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), ...
      @(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_vfemp1.Loperator(4,4,{1,0,0,0;0,1,0,0;0,0,1,0;0,0,0,1},[],[],[]);
pde=fc_vfemp1.PDE(Lop,f);
bvp=fc_vfemp1.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	0.230 (s)	0.043 (s)
7	4 096	57 624	0.411 (s)	0.551 (s)
9	10 000	157 464	0.955 (s)	4.715 (s)
11	20 736	351 384	2.359 (s)	24.537 (s)
13	38 416	685 464	4.603 (s)	117.486 (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.

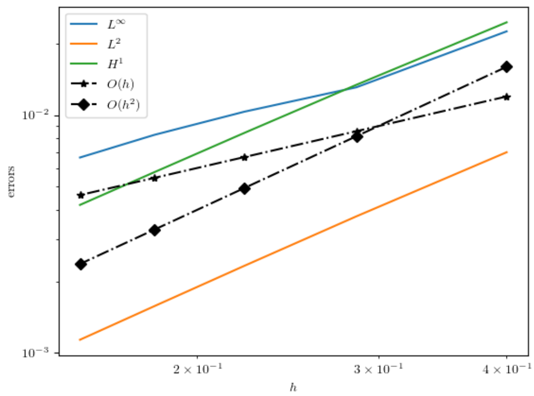


Figure 3.6: 4D Poisson BVP with mixed boundary conditions: order of the P_1-Lagrange finite element method in function of the mesh size h for meshes generated with the fc_simesh.hypercube function.

remark 3.4

In the vfem$\mathbb{P}_1$ Octave package 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:

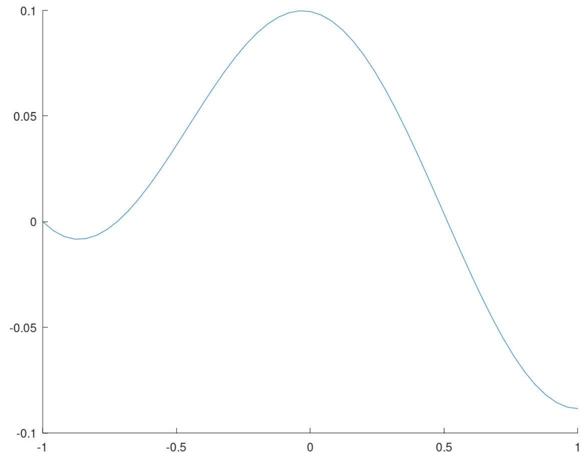
```
setBVP=@(N,verb)fc_vfemp1.examples.Poisson.setBVPPoisson4D_ex01(N,verb);
fc_vfemp1.examples.bench('LN',5:2:15,'setBVP',setBVP);
```

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

$$
\begin{aligned}
-\operatorname{div}(\alpha \nabla u) + \langle \boldsymbol{V}, \nabla u \rangle + \beta u &= f \quad \text{in } \Omega \subset \mathbb{R}^2, && (3.23)\\
u &= 4 \quad \text{on } \Gamma_2, && (3.24)\\
u &= -4 \quad \text{on } \Gamma_4, && (3.25)\\
u &= 0 \quad \text{on } \Gamma_{20} \cup \Gamma_{21}, && (3.26)\\
\frac{\partial u}{\partial n} &= 0 \quad \text{on } \Gamma_1 \cup \Gamma_3 \cup \Gamma_{10} && (3.27)
\end{aligned}
$$

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

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

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

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

where

- $\mathcal{L} := \mathcal{L}_{\alpha\mathbb{I}, \mathbf{0}, \boldsymbol{V}, \beta}$, and then the conormal derivative of u is given by
$$\frac{\partial u}{\partial n_{\mathcal{L}}} := \langle \mathbb{A} \nabla u, \boldsymbol{n} \rangle - \langle \boldsymbol{b} u, \boldsymbol{n} \rangle = \alpha \frac{\partial u}{\partial n}.$$
- $\Gamma^D = \Gamma_2 \cup \Gamma_4 \cup \Gamma_{20} \cup \Gamma_{21}$ and $\Gamma^R = \Gamma_1 \cup \Gamma_3 \cup \Gamma_{10}$
- $g^D := 4$ on Γ_2, and $g^D := -4$ on Γ_4 and $g^D := 0$ on $\Gamma_{20} \cup \Gamma_{21}$

- $a^R = g^R := 0$ on Γ^R.

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

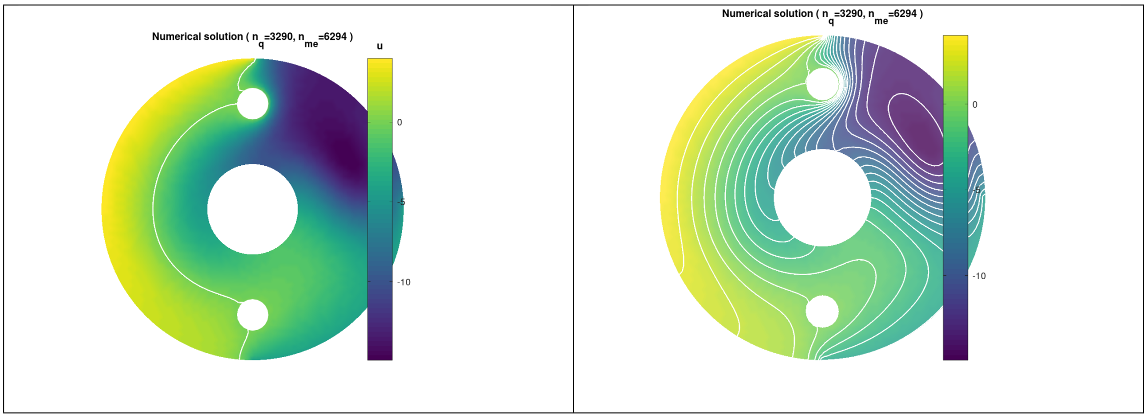
Algorithm 1 Stationary convection-diffusion problem in 2D

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

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 package 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.




```
geofile='disk3holes';
fullgeofile=fc_vfemp1.get_geo(2,2,geofile);
af=@(x,y) 0.1+y.^2;
Vx=@(x,y)  10*y;Vy=@(x,y) 10*x;
b=0.01;g2=4;g4= 4;
f=@(x,y)  200.0*exp( ((x 0.75).^2+y.^2)/(0.1));
meshfile=fc_oogmsh.gmsh.buildmesh2d(fullgeofile,N,'verbose',verbose,varargin{:});
Th=fc_simesh.siMesh(meshfile);
Lop=fc_vfemp1.Loperator(Th.dim,Th.d,{af,[];[],af},[],{Vx,Vy},b);
pde=fc_vfemp1.PDE(Lop,f);
bvp=fc_vfemp1.BVP(Th,pde);
bvp.setDirichlet(2, g2);
bvp.setDirichlet(4, g4);
bvp.setDirichlet(20, 0.);
bvp.setDirichlet(21, 0.);
```

Listing 3.6: Setting the 2D stationary convection-diffusion BVP and representation of the numerical solution. Part of the file `+fc_vfemp1/+examples/setBVPStationaryConvectionDiffusion2D01.m`

N	n_q	n_{me}	Assembly	Solve
100	33 503	66 060	0.164 (s)	0.238 (s)
200	131 165	260 444	0.481 (s)	1.147 (s)
300	294 356	585 882	1.391 (s)	3.071 (s)
400	521 219	1 038 668	2.688 (s)	5.975 (s)
500	814 835	1 624 954	4.594 (s)	10.198 (s)
600	1 169 219	2 332 782	7.204 (s)	15.664 (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 fc-vfemP1 Octave package 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 $\mathcal{C}^r_A([z_{min}, z_{max}])$ be the right circular cylinder along $z-$axis ($z \in [z_{min}, z_{max}]$) with bases the circles of radius r and center (x_A, y_A, z_{min}) and (x_A, y_A, z_{max}). Let Ω be the cylinder defined by

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

We respectively denote by Γ_{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 $\mathcal{C}^1_{(0,0)}([0,3])$, $\mathcal{C}^{0.3}_{(0,0)}([0,3])$, $\mathcal{C}^{0.1}_{(0,-0.7)}([0,3])$ and $\mathcal{C}^{0.1}_{(0,0.7)}([0,3])$. The domain Ω and its boundaries are represented in Figure 3.8.

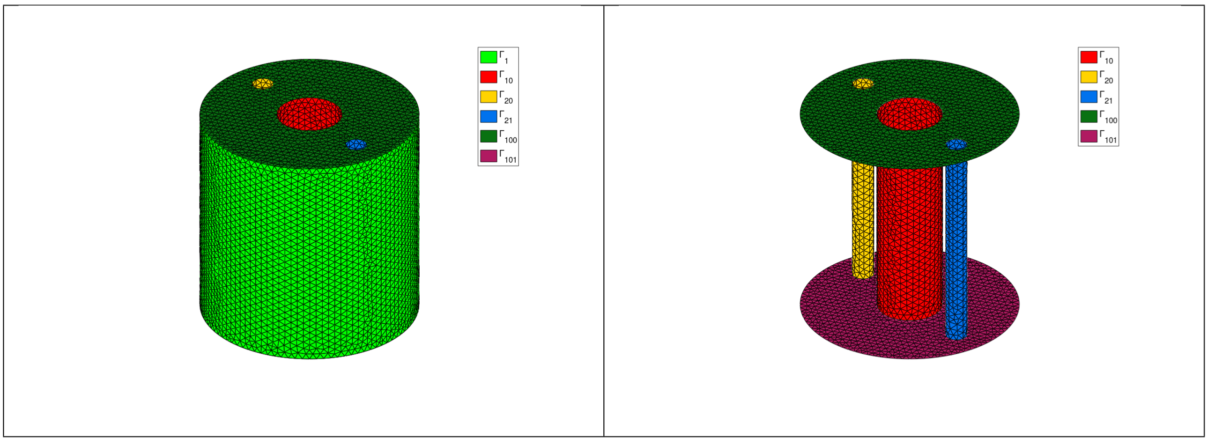


Figure 3.8: 3D stationary convection-diffusion BVP : all boundaries (left) and boundaries without Γ_1 (right)

The 3D problem to solve is the following

***Usual* BVP 4 :**

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

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

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

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

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

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

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

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

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

where

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

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

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

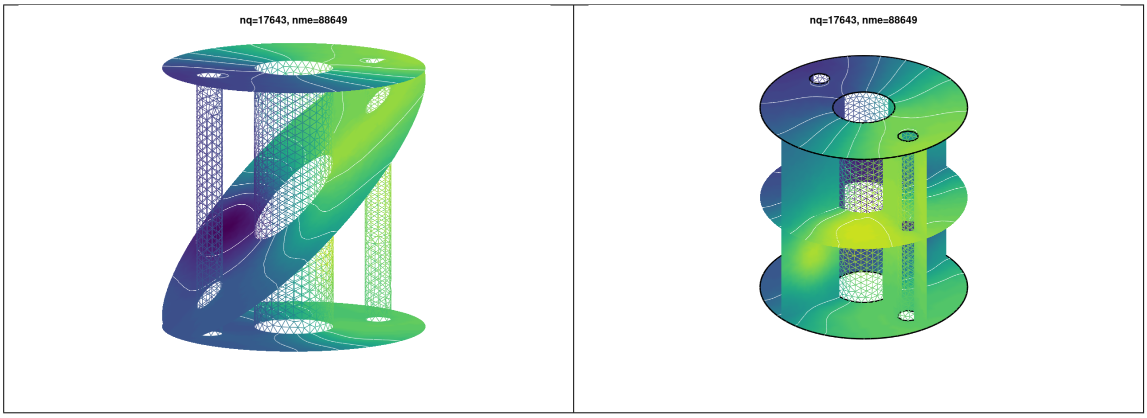
$$a^R = \begin{cases} 0 & \text{on } \Gamma_1 \cup \Gamma_{10} \cup \Gamma_{100} \cup \Gamma_{101} \\ 1 & \text{on } \Gamma_{20} \cup \Gamma_{21} \end{cases}$$

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

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 package 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.




```
geofile='cylinder3holes';
fullgeofile=fc_vfemp1.get_geo(3,3,geofile);
af=@(x,y,z) 0.7+ z/10;
beta=0.01;
V={@(x,y,z) 10*y ,@(x,y,z) 10*x ,@(x,y,z) 10*z};
f=@(x,y,z) 800*exp( 10*((x 0.65).^2+y.^2+(z 0.5).^2)) + 800*exp( 10*((x+0.65).^2+y.^2+(z 0.5).^2));
meshfile=fc_oogmsh.gmsh.buildmesh3d(fullgeofile,N,'verbose',verbose,varargin{:});
Th=fc_simesh.siMesh(meshfile,'dim',3);
Lop=fc_vfemp1.Loperator(Th.dim,Th.d,{af,[],[];[],af,[];[],[],af},[],V,beta);
pde=fc_vfemp1.PDE(Lop,f);
bvp=fc_vfemp1.BVP(Th,pde);
bvp.setRobin(20,0.05,1);
bvp.setRobin(21, 0.05,1);
```

Listing 3.7: Setting the 3D stationary convection-diffusion BVP and representations of the numerical solution. Part of the file `+fc_vfemp1/+examples/setBVPStationaryConvectionDiffusion3D01.m`

N	$\mathrm{n_q}$	$\mathrm{n_{me}}$	Assembly	Solve
15	17 643	88 649	0.293 (s)	0.656 (s)
20	38 614	204 301	0.642 (s)	2.296 (s)
25	72 245	395 317	1.301 (s)	5.839 (s)
30	120 802	674 231	2.674 (s)	14.393 (s)
35	186 763	1 061 295	4.464 (s)	32.056 (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 Octave package 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 $\boldsymbol{j}$ be the current density in A/m^2. It's satisfy

$$\operatorname{div} \boldsymbol{j} = 0, \quad \text{in } \Omega. \tag{3.32}$$

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

where σ is the local electrical conductivity and $\boldsymbol{E}$ the local electric field.
The electric field can be written as a gradient of a scalar potential

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

Combining all these equations leads to Laplace's equation

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

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 $20°C$	$\sigma(S/m)$ at $20°C$
Carbon (graphene)	1.00×10^{-8}	1.00×10^{8}
Gold	2.44×10^{-8}	4.10×10^{8}
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

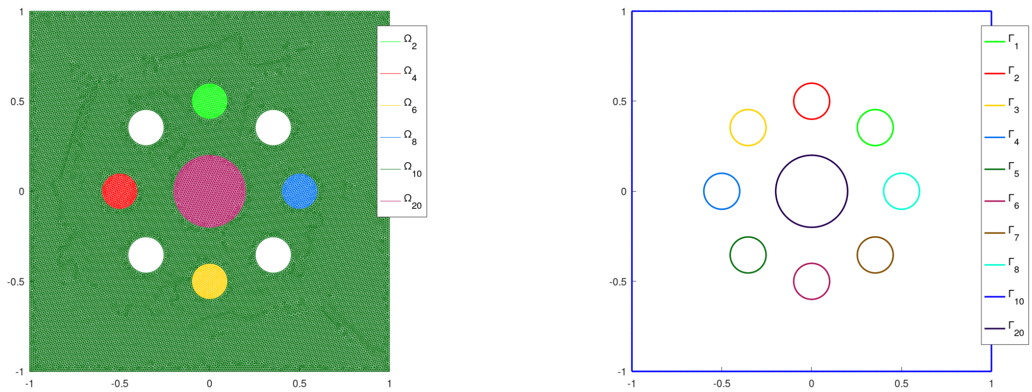


Figure 3.9: Mesh from `square4holes6dom.geo`, domains representation (left) and boundaries (right)

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

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

We solve the following BVP

Usual BVP 5 : 2D electrostatic problem

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

$$\operatorname{div}(\sigma \boldsymbol{\nabla}\varphi) = 0 \quad \text{in } \Omega, \tag{3.36}$$

$$\varphi = 0 \quad \text{on } \Gamma_3 \cup \Gamma_7, \tag{3.37}$$

$$\varphi = 12 \quad \text{on } \Gamma_1 \cup \Gamma_5, \tag{3.38}$$

$$\sigma \frac{\partial \varphi}{\partial n} = 0 \quad \text{on } \Gamma_{10}. \tag{3.39}$$

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

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

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

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

To write this problem properly with the fc vfem$\mathbb{P}_1$ package we split (3.36) in two parts

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

and we set these PDEs on each domains. This is done in 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  > 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 $\boldsymbol{E}$. Computational times for assembling and solving steps are given in Table 3.17.

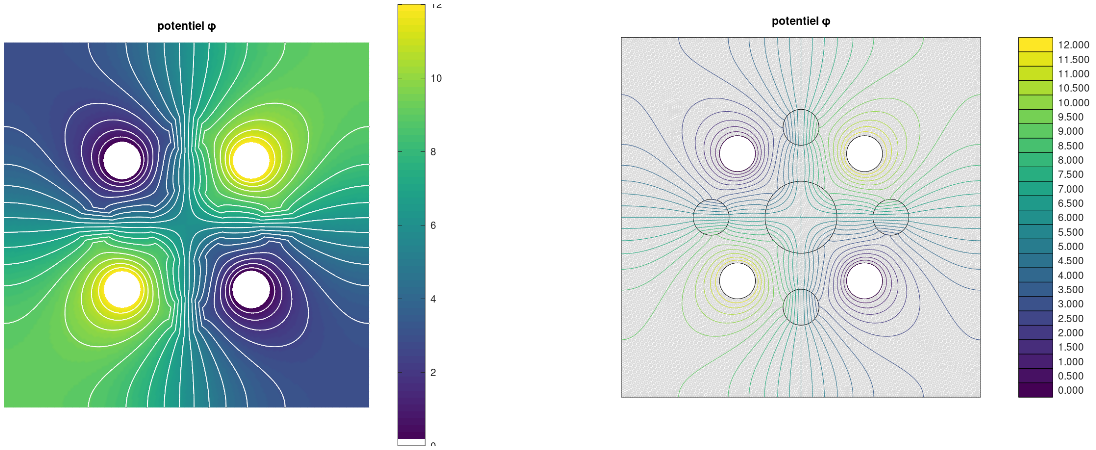


Figure 3.10: Test 1, potential φ

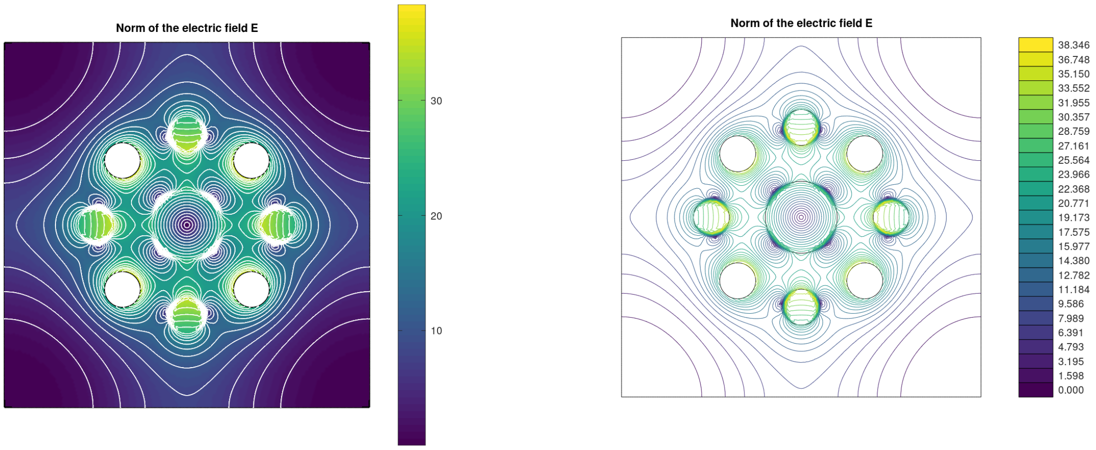


Figure 3.11: Test 1, norm of the electrical field $\boldsymbol{E}$

N	n_q	n_{me}	Assembly	Solve
100	46 731	92 412	0.196 (s)	0.348 (s)
150	104 604	207 630	0.333 (s)	0.932 (s)
200	184 691	367 276	0.563 (s)	1.778 (s)
250	288 466	574 298	0.890 (s)	3.007 (s)
300	414 577	825 992	1.403 (s)	4.669 (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 fc vfem$\mathbb{P}_1$ Octave package 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 [?]).

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

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

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

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

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

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

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

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

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

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

The problem to solve is the following

Usual vector BVP 2 : Elasticity problem

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

$$\begin{aligned} -\operatorname{div}(\boldsymbol{\sigma}(\boldsymbol{u})) &= \boldsymbol{f}, \quad \text{in } \Omega \subset \mathbb{R}^d, & (4.1)\\ \boldsymbol{\sigma}(\boldsymbol{u}).\boldsymbol{n} &= \boldsymbol{0} \quad \text{on } \Gamma^R, & (4.2)\\ \boldsymbol{u} &= \boldsymbol{0} \quad \text{on } \Gamma^D. & (4.3) \end{aligned}$$

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 $\mathcal{H}_{\alpha,\beta} = \mathcal{L}_{\mathbb{A}^{\alpha,\beta},\boldsymbol{0},\boldsymbol{0},0}$, $\forall(\alpha,\beta) \in [\![1,d]\!]^2$, with

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

then

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

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

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

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 $\forall(\alpha,\beta) \in [\![1,d]\!]^2$, $\mathcal{H}_{\alpha,\beta} = \mathcal{L}_{\mathbb{A}^{\alpha,\beta},\boldsymbol{0},\boldsymbol{0},0}$, with

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

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

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

$$\begin{aligned} \mathcal{H}(\boldsymbol{u}) &= \boldsymbol{f}, && \text{in } \Omega, & (4.7)\\ \frac{\partial \boldsymbol{u}}{\partial n_{\mathcal{H}_\alpha}} &= 0, && \text{on } \Gamma^R_\alpha = \Gamma^R, \ \forall \alpha \in [\![1,d]\!] & (4.8)\\ \boldsymbol{u}_\alpha &= 0, && \text{on } \Gamma^D_\alpha = \Gamma^D, \ \forall \alpha \in [\![1,d]\!]. & (4.9) \end{aligned}$$

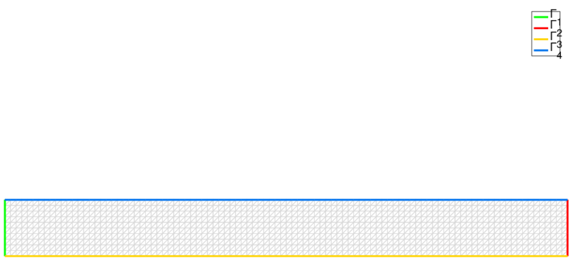


Figure 4.1: Domain and boundaries

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, Lame's coefficients verify

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

The material is rubber so $E = 21\,.10^5$Pa and $\nu = 0.45$. We also have $\boldsymbol{f} = \boldsymbol{x} \mapsto (0,-1)^t$ and we choose $\Gamma^R = \Gamma^2 \cup \Gamma^3 \cup \Gamma^4$, $\Gamma^D = \Gamma^2$.

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

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

$$\frac{\partial \boldsymbol{u}}{\partial n_{\mathcal{H}_\alpha}} = 0, \quad \text{on } \Gamma^R_\alpha = \Gamma^2 \cup \Gamma^3 \cup \Gamma^4,\ \forall \alpha \in [\![1,2]\!] \tag{4.11}$$

$$\boldsymbol{u}_\alpha = 0, \quad \text{on } \Gamma^D_\alpha = \Gamma^1,\ \forall \alpha \in [\![1,2]\!]. \tag{4.12}$$

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

Listing 4.1: **2D elasticity, Octave 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 Octave 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.

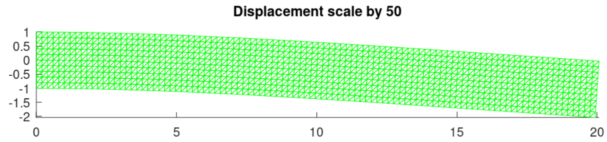


Figure 4.2: Mesh displacement scaled by a factor 50 for the 2D elasticity problem

N	n_q	n_{me}	n_{dof}	Assembly	Solve
100	101 101	200 000	202 202	0.501 (s)	2.716 (s)
150	226 651	450 000	453 302	1.197 (s)	6.734 (s)
200	402 201	800 000	804 402	2.768 (s)	14.081 (s)
250	627 751	1 250 000	1 255 502	4.673 (s)	22.478 (s)
300	903 301	1 800 000	1 806 602	7.442 (s)	34.777 (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 vfem$\mathbb{P}_1$ Octave package 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.

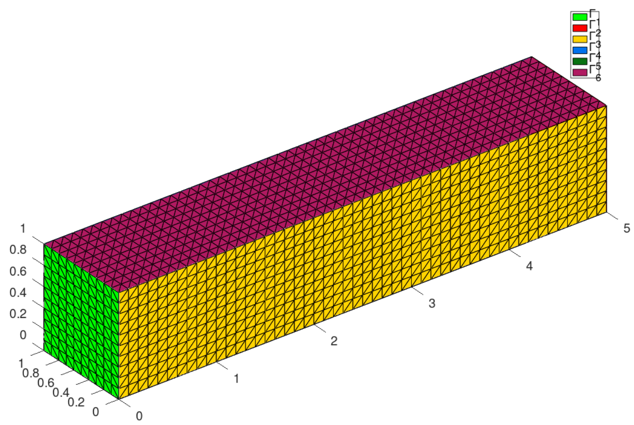


Figure 4.3: Domain and boundaries

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

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

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

$$\frac{\partial \boldsymbol{u}}{\partial n_{\mathcal{H}_\alpha}} = 0, \quad \text{on } \Gamma_\alpha^R = \bigcup_{i=2}^{6} \Gamma_i, \ \forall \alpha \in [\![1, 3]\!] \tag{4.14}$$

$$\boldsymbol{u}_\alpha = 0, \quad \text{on } \Gamma_\alpha^D = \Gamma^1, \ \forall \alpha \in [\![1, 3]\!]. \tag{4.15}$$

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

Listing 4.2: **3D elasticity, Octave 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.

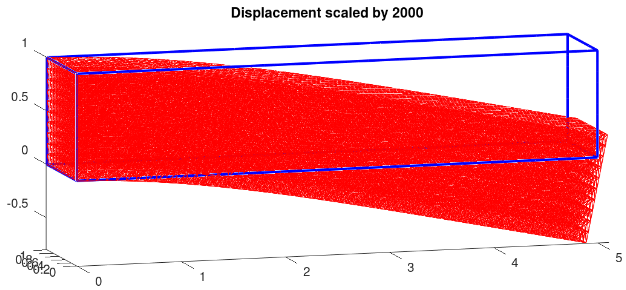


Figure 4.4: Result for the 3D elasticity problem

N	$\mathrm{n_q}$	$\mathrm{n_{me}}$	$\mathrm{n_{dof}}$	Assembly	Solve
10	6 171	30 000	18 513	0.247 (s)	0.871 (s)
15	19 456	101 250	58 368	0.807 (s)	5.275 (s)
20	44 541	240 000	133 623	2.354 (s)	18.828 (s)
25	85 176	468 750	255 528	5.703 (s)	61.808 (s)
30	145 111	810 000	435 333	10.205 (s)	148.822 (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 vfem$\mathbb{P}_1$ Octave package 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

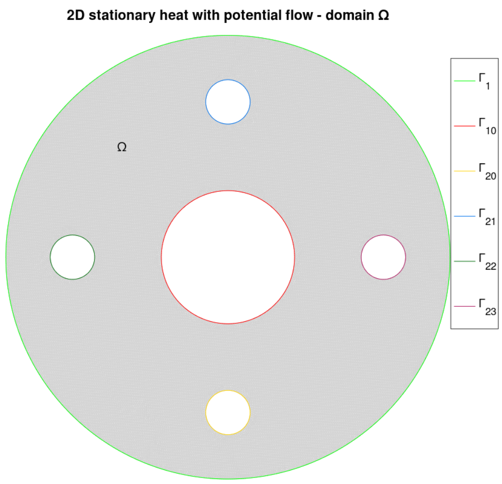


Figure 4.5: Domain and boundaries

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

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

$$
\begin{aligned}
-\operatorname{div}(\alpha \boldsymbol{\nabla} u) + \langle \boldsymbol{V}, \boldsymbol{\nabla} u \rangle + \beta u &= 0 \ \text{ in } \Omega \subset \mathbb{R}^2, && (4.16)\\
u &= 20 * \boldsymbol{x}_2 \ \text{ on } \Gamma_{21}, && (4.17)\\
u &= 0 \ \text{ on } \Gamma_{22} \cup \Gamma_{23}, && (4.18)\\
\frac{\partial u}{\partial n} &= 0 \ \text{ on } \Gamma_1 \cup \Gamma_{10} \cup \Gamma_{20} && (4.19)
\end{aligned}
$$

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

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

The potential flow is the velocity field $\boldsymbol{V} = \boldsymbol{\nabla} \phi$ 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 $\phi \in \mathrm{H}^2(\Omega)$ such that

$$
\begin{aligned}
-\Delta \phi &= 0 \ \text{ in } \Omega, && (4.20)\\
\phi &= -20 \ \text{ on } \Gamma_{21}, && (4.21)\\
\phi &= 20 \ \text{ on } \Gamma_{20}, && (4.22)\\
\frac{\partial \phi}{\partial n} &= 0 \ \text{ on } \Gamma_1 \cup \Gamma_{23} \cup \Gamma_{22} && (4.23)
\end{aligned}
$$

Then the potential flow $\boldsymbol{V}$ is *solution* of (4.24)

Usual vector BVP 3 : 2D potential flow

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

$$
\boldsymbol{V} = \boldsymbol{\nabla} \phi \ \text{ in } \Omega, \qquad (4.24)
$$

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

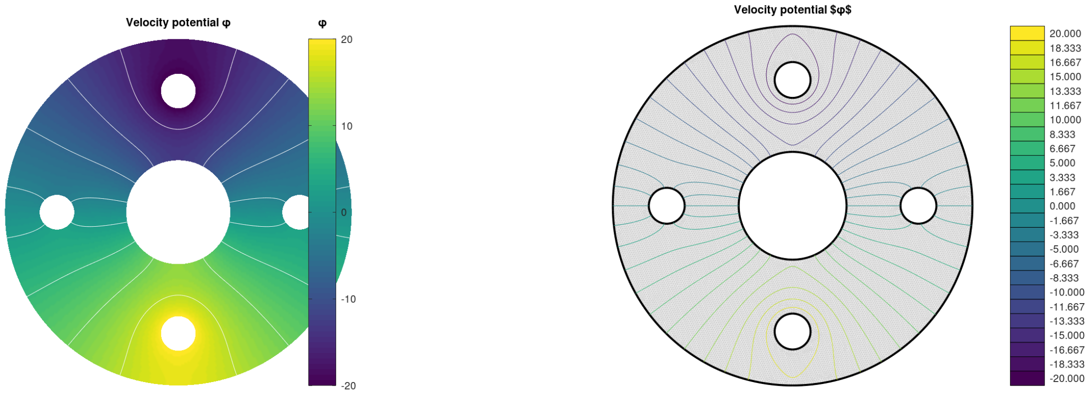


Figure 4.6: Velocity potential Φ

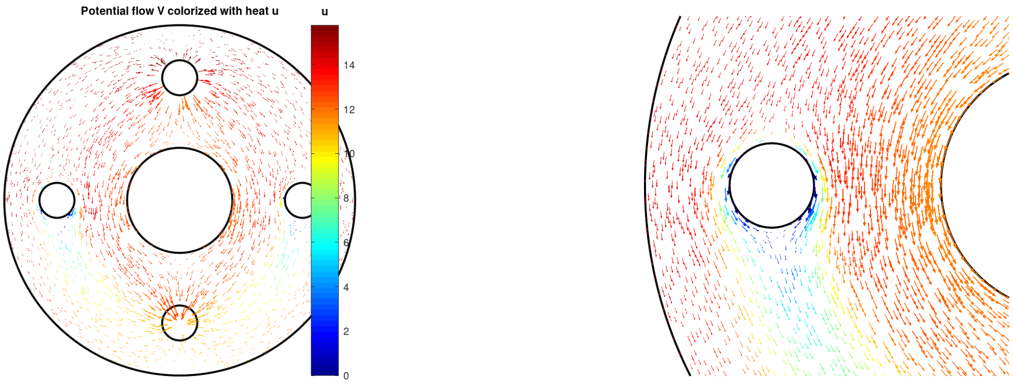


Figure 4.7: Potential flow $\boldsymbol{V} = \boldsymbol{\nabla}\,\phi$ colored with u heat values

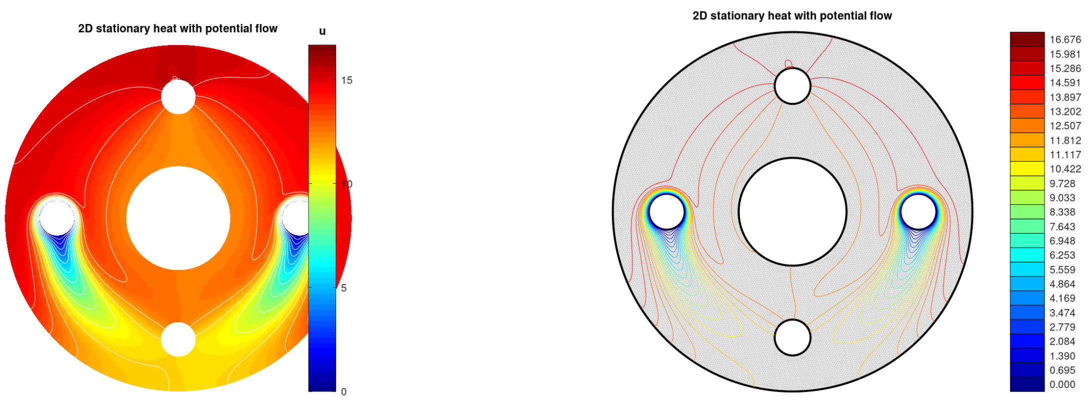


Figure 4.8: Heat u with potential flow $\boldsymbol{V}$

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

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

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

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

- $\Gamma^D = \Gamma_{20} \cup \Gamma_{21}$ 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 package to solve (4.20)-(4.23) is given in Listing 4.7.

Listing 4.3: **Stationary heat with potential flow in 2D, Octave 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 $\boldsymbol{V}$, we can write the potential flow problem (4.24) with $\mathcal{H}$-operators as

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

where

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

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

Listing 4.4: **Stationary heat with potential flow in 2D, Octave 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 $\boldsymbol{V}_1$ and $\boldsymbol{V}_2$.

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

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

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

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

The code using the package fc-vfem$\mathbb{P}_1$ to solve *Scalar* BVP 11 is given in Listing 4.7.

Listing 4.5: **Stationary heat with potential flow in 2D, Octavecode (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 $\phi \in \mathrm{H}^2(\Omega)$ and $\boldsymbol{V} = (\boldsymbol{V}_1, \boldsymbol{V}_2) \in \mathrm{H}^1(\Omega) \times \mathrm{H}^1(\Omega)$ such that

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

$$\boldsymbol{V}_1 - \frac{\partial \phi}{\partial x} = 0 \quad \text{in } \Omega, \tag{4.26}$$

$$\boldsymbol{V}_2 - \frac{\partial \phi}{\partial y} = 0 \quad \text{in } \Omega, \tag{4.27}$$

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

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

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

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

Let $\boldsymbol{w} = \begin{pmatrix}\phi\\ \boldsymbol{V}_1\\ \boldsymbol{V}_2\end{pmatrix}$, 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 $\boldsymbol{w} = (\boldsymbol{w}_1, \boldsymbol{w}_2, \boldsymbol{w}_3) \in (\mathrm{H}^2(\Omega))^3$ such that

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

$$\boldsymbol{w}_\alpha = g^D_\alpha \quad \text{on } \Gamma^D_\alpha,\ \forall \alpha \in [\![1,3]\!], \tag{4.32}$$

$$\frac{\partial \boldsymbol{w}}{\partial n_{\mathcal{H}_\alpha}} = g^R_\alpha \quad \text{on } \Gamma^R_\alpha,\ \forall \alpha \in [\![1,3]\!], \tag{4.33}$$

where

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

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

- $\Gamma^R_\alpha = \Gamma^D_\alpha = \emptyset$ for all $\alpha \in \{2, 3\}$ (no boundary conditions on $\boldsymbol{V}_1$ and $\boldsymbol{V}_2$)
- $\Gamma^D_1 = \Gamma_{20} \cup \Gamma_{21}$ and $\Gamma^R_1 = \Gamma_1 \cup \Gamma_{10} \cup \Gamma_{22} \cup \Gamma_{23}$
- $g^D_1 := 20$ on Γ_{20}, and $g^D_1 := -20$ on Γ_{21}

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

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

So we obtain

$$\frac{\partial \boldsymbol{w}}{\partial n_{\mathcal{H}_1}} \stackrel{\mathsf{def}}{=} \sum_{\alpha=1}^{3} \frac{\partial \boldsymbol{w}_\alpha}{\partial n_{\mathcal{H}_{1,\alpha}}} = \langle \boldsymbol{V}, \boldsymbol{n} \rangle = \frac{\partial \phi}{\partial \boldsymbol{n}}, \tag{4.34}$$

and

$$\frac{\partial \boldsymbol{w}}{\partial n_{\mathcal{H}_2}} = \frac{\partial \boldsymbol{w}}{\partial n_{\mathcal{H}_3}} := 0. \tag{4.35}$$

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

By using the package fc vfem$\mathbb{P}_1$ the Octave 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, Octave 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));
Hop.set(3,1,fc_vfemp1.Loperator(d,d,[],[],{0, 1},[]));
Hop.set(3,3,fc_vfemp1.Loperator(d,d,[],[],[],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 package fc vfem$\mathbb{P}_1$ for solving (4.16)-(4.19) is given in Listing 4.6

Listing 4.7: **Stationary heat with potential flow in 2D, Octave 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.

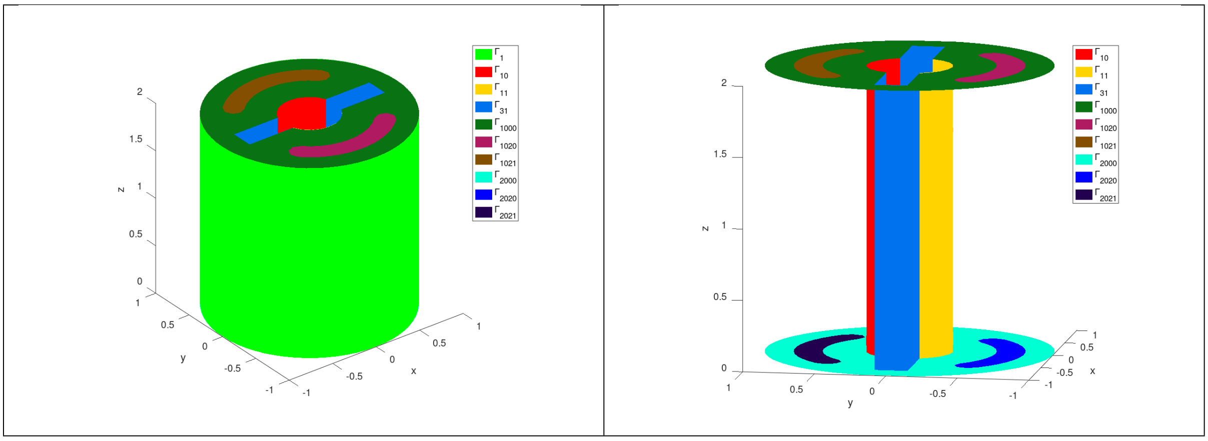


Figure 4.9: Stationary heat with potential flow : 3d mesh

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

$$\begin{aligned} -\operatorname{div}(\alpha \boldsymbol{\nabla} u) + \langle \boldsymbol{V}, \boldsymbol{\nabla} u\rangle + \beta u &= 0 \quad \text{in } \Omega \subset \mathbb{R}^3, && (4.36)\\ u &= 30 \quad \text{on } \Gamma_{1020}\cup\Gamma_{2020}, && (4.37)\\ u &= 10\delta_{|z-1|>0.5} \quad \text{on } \Gamma_{10}, && (4.38)\\ \frac{\partial u}{\partial n} &= 0 \quad \text{otherwise} && (4.39)\end{aligned}$$

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

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

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

***Usual* BVP 9 : 3D velocity potential**

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

$$\begin{aligned} -\Delta\phi &= 0 \quad \text{in } \Omega, && (4.40)\\ \phi &= 1 \quad \text{on } \Gamma_{1021}\cup\Gamma_{2021}, && (4.41)\\ \phi &= -1 \quad \text{on } \Gamma_{1020}\cup\Gamma_{2020}, && (4.42)\\ \frac{\partial\phi}{\partial n} &= 0 \quad \text{otherwise} && (4.43)\end{aligned}$$

Then the potential flow $\boldsymbol{V}$ is *solution* of (4.44)

***Usual* vector BVP 5 : 3D potential flow**

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

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

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

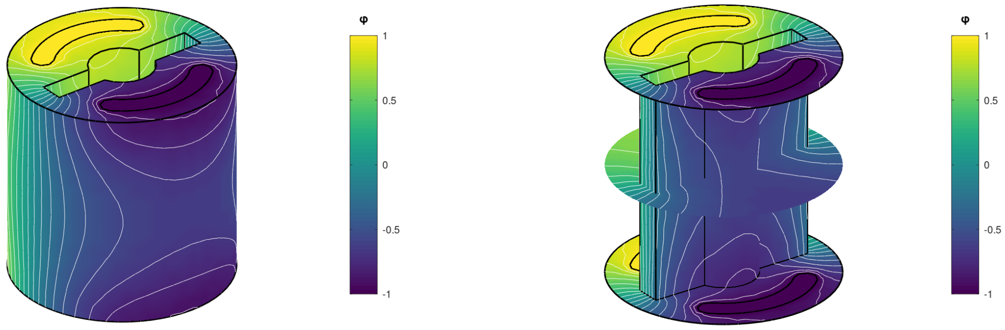


Figure 4.10: Velocity potential ϕ

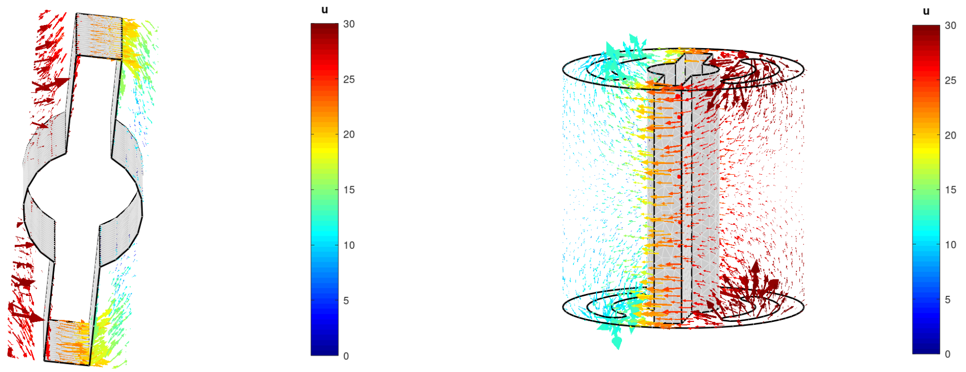


Figure 4.11: Potential flow $\boldsymbol{V} = \boldsymbol{\nabla}\,\phi$ colored with u heat values

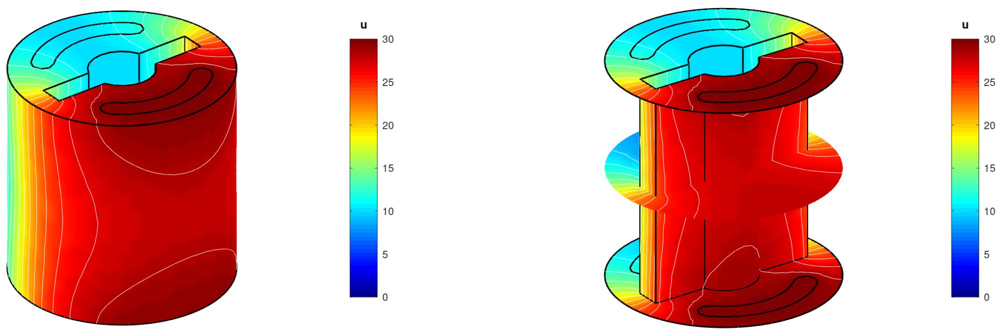


Figure 4.12: Heat solution u

Now we will present two ways to solve these problems using the vfem$\mathbb{P}_1$ Octave package.

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

- $\mathcal{L} := \mathcal{L}_{\mathbb{I},\mathbf{0},\mathbf{0},0}$, and then the conormal derivative of ϕ is given by
$$\frac{\partial \phi}{\partial n_{\mathcal{L}}} := \langle \mathbb{A}\,\boldsymbol{\nabla}\,\phi, \boldsymbol{n}\rangle - \langle \boldsymbol{b}\phi, \boldsymbol{n}\rangle = \frac{\partial \phi}{\partial n}.$$
- $\Gamma^D = \Gamma_{1020} \cup \Gamma_{1021} \cup \Gamma_{2020} \cup \Gamma_{2021}$
- $g^D := 1$ 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 $\mathsf{vfem}\mathbb{P}_1$ package 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, Octave 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 $\boldsymbol{V}$, we can write the potential flow problem (4.44)

- with $\mathcal{H}$-operators as
$$\boldsymbol{V} = \begin{pmatrix}\boldsymbol{V}_1\\ \boldsymbol{V}_2\\ \boldsymbol{V}_2\end{pmatrix} = \mathcal{B}\begin{pmatrix}\phi\\ \phi\\ \phi\end{pmatrix}$$
where
$$\mathcal{B} = \begin{pmatrix}\mathcal{L}_{\mathbb{O}_3,\mathbf{0}_3,(1,0,0)^t,1} & 0 & 0\\ 0 & \mathcal{L}_{\mathbb{O}_3,\mathbf{0}_3,(0,1,0)^t,0} & 0\\ 0 & 0 & \mathcal{L}_{\mathbb{O}_3,\mathbf{0}_3,(0,0,1)^t,0}\end{pmatrix}$$
- with $\mathcal{L}$-operators as
$$\boldsymbol{V} = \begin{pmatrix}\boldsymbol{V}_1\\ \boldsymbol{V}_2\\ \boldsymbol{V}_2\end{pmatrix} = \boldsymbol{\nabla}\,\phi = \begin{pmatrix}\mathcal{L}_{\mathbb{O}_3,\mathbf{0}_3,(1,0,0)^t,0}(\phi)\\ \mathcal{L}_{\mathbb{O}_3,\mathbf{0}_3,(0,1,0)^t,0}(\phi)\\ \mathcal{L}_{\mathbb{O}_3,\mathbf{0}_3,(0,0,1)^t,0}(\phi)\end{pmatrix}$$

The code using the $\mathsf{vfem}\mathbb{P}_1$ package for solving this problem with $\mathcal{L}$-operators is given in Listing 4.9.

Listing 4.9: **Stationary heat with potential flow in 3D, Octave 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 $u \in \mathrm{H}^1(\Omega)$ 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

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

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

- $\Gamma^D = \Gamma_{1020} \cup \Gamma_{2020} \cup \Gamma_{10}$
- $\Gamma^R = \Gamma_1 \cup \Gamma_{11} \cup \Gamma_{31} \cup \Gamma_{1000} \cup \Gamma_{1021} \cup \Gamma_{2000} \cup \Gamma_{2021}$
- $g^D(x, y, z) := 30$ 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 fc vfem$\mathbb{P}_1$ package is given in Listing 4.10.

Listing 4.10: **Stationary heat with potential flow in 3D, Octave 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 $\boldsymbol{V}$. For that we can rewrite the potential flow problem (4.40)-(4.43), by introducing $\boldsymbol{V} = (\boldsymbol{V}_1, \boldsymbol{V}_2, \boldsymbol{V}_3)$ as unknowns :

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

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

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

$$\boldsymbol{V}_1 - \frac{\partial \phi}{\partial x} = 0 \text{ in } \Omega, \tag{4.46}$$

$$\boldsymbol{V}_2 - \frac{\partial \phi}{\partial y} = 0 \text{ in } \Omega, \tag{4.47}$$

$$\boldsymbol{V}_3 - \frac{\partial \phi}{\partial z} = 0 \text{ in } \Omega, \tag{4.48}$$

with boundary conditions (4.41) to (4.43).

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

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

$$\mathcal{H}(\boldsymbol{w}) = 0 \tag{4.49}$$

with

$$\mathcal{H}_{1,1} = 0, \quad \mathcal{H}_{1,2} = \mathcal{L}_{\mathbb{O},-\boldsymbol{e}_1,\boldsymbol{0},0}, \quad \mathcal{H}_{1,3} = \mathcal{L}_{\mathbb{O},-\boldsymbol{e}_2,\boldsymbol{0},0}, \quad \mathcal{H}_{1,4} = \mathcal{L}_{\mathbb{O},-\boldsymbol{e}_3,\boldsymbol{0},0}, \tag{4.50}$$

$$\mathcal{H}_{2,1} = \mathcal{L}_{\mathbb{O},\boldsymbol{0},-\boldsymbol{e}_1,0}, \quad \mathcal{H}_{2,2} = \mathcal{L}_{\mathbb{O},\boldsymbol{0},\boldsymbol{0},1}, \quad \mathcal{H}_{2,3} = 0, \quad \mathcal{H}_{2,4} = 0, \tag{4.51}$$

$$\mathcal{H}_{3,1} = \mathcal{L}_{\mathbb{O},\boldsymbol{0},-\boldsymbol{e}_2,0}, \quad \mathcal{H}_{3,2} = 0, \quad \mathcal{H}_{3,3} = \mathcal{L}_{\mathbb{O},\boldsymbol{0},\boldsymbol{0},1}, \quad \mathcal{H}_{3,4} = 0, \tag{4.52}$$

$$\mathcal{H}_{4,1} = \mathcal{L}_{\mathbb{O},\boldsymbol{0},-\boldsymbol{e}_3,0}, \quad \mathcal{H}_{4,2} = 0, \quad \mathcal{H}_{4,3} = 0, \quad \mathcal{H}_{4,4} = \mathcal{L}_{\mathbb{O},\boldsymbol{0},\boldsymbol{0},1}, \tag{4.53}$$

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

The conormal derivatives are given by

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

So we obtain

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

and

$$\sum_{\alpha=1}^{4} \frac{\partial \boldsymbol{w}_\alpha}{\partial n_{\mathcal{H}_{2,\alpha}}} = \sum_{\alpha=1}^{4} \frac{\partial \boldsymbol{w}_\alpha}{\partial n_{\mathcal{H}_{3,\alpha}}} = \sum_{\alpha=1}^{4} \frac{\partial \boldsymbol{w}_\alpha}{\partial n_{\mathcal{H}_{4,\alpha}}} = 0. \tag{4.55}$$

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

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

$$\boldsymbol{w}_\alpha = g_\alpha^D \quad \text{on } \Gamma_\alpha^D,\ \forall \alpha \in [\![1,4]\!], \tag{4.57}$$

$$\frac{\partial \boldsymbol{w}}{\partial n_{\mathcal{H}_\alpha}} = 0 \quad \text{on } \Gamma_\alpha^R,\ \forall \alpha \in [\![1,4]\!], \tag{4.58}$$

where

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

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

- $\Gamma_\alpha^R = \Gamma_\alpha^D = \emptyset$ for all $\alpha \in \{2,3,4\}$ (no boundary conditions on $\boldsymbol{V}_1$, $\boldsymbol{V}_2$ and $\boldsymbol{V}_3$)
- $\Gamma_1^D = \Gamma_{1020} \cup \Gamma_{1021} \cup \Gamma_{2020} \cup \Gamma_{2021}$ and $\Gamma_1^R = \Gamma \setminus \Gamma_.^D$
- $g_1^D = \delta_{\Gamma_{1020} \cup \Gamma_{2020}} - \delta_{\Gamma_{1021} \cup \Gamma_{2021}}$,

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

Listing 4.11: **Stationary heat with potential flow in 3D, Octave 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 describded in previous section. The code is given in Listing 4.12

Listing 4.12: **Stationary heat with potential flow in 3D, Octave 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 fc vfem$\mathbb{P}_1$ Octave package add-on tools are provided. They allow to easily distribute codes using the package. 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

$$-(\operatorname{div}\sigma(\boldsymbol{u}))_i \quad = \quad f_i,\ \forall i \in [\![1,d]\!],\ \text{in } \Omega \tag{1}$$

We have

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

and

$$\epsilon_{ij}(\boldsymbol{u}) = \frac{1}{2}\left(\frac{\partial \boldsymbol{u}_i}{\partial x_j} + \frac{\partial \boldsymbol{u}_j}{\partial x_i}\right)$$

So we obtain

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

So, from (1.18) and (1) we want $\forall i \in [\![1,d]\!]$

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

and by identification, we obtain $\forall i \in [\![1,d]\!]$

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

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

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

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

$$\sum_{j=1}^{d}\operatorname{div}(\mathbb{A}^{i,j}\,\boldsymbol{\nabla}\,\boldsymbol{u}_j) + f_i = 0,\ \forall i \in [\![1,d]\!],\ \text{in } \Omega \tag{2}$$

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

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

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

and in dimension $d=3$

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

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

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

Boundary conditions

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

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

But we have

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

and, for $j\neq i$

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

So we obtain

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

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

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

with

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

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

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

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

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

So we have proved that

$$(\sigma(\vec{u})\boldsymbol{n})_i = \sum_{j=1}^{d}\frac{\partial\boldsymbol{u}_j}{\partial n_{i,j}},\ \forall i\in[\![1,d]\!] \tag{9}$$

Listings

[1] F. Cuvelier. fc_oogmsh: an object-oriented Octabe package 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 Octabe package 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 vfem$\mathbb{P}_1$ Octave package

git informations on the packages used to build this manual

```
-------------------------------------------------
   name : fc-vfemp1
    tag : 0.2.1
 commit : 0be88b947ad32ccf9e216ee1f0921ee3bcbcb69b
   date : 2020-03-19
   time : 06-37-11
 status : 0
-------------------------------------------------
   name : fc-tools
    tag : 0.0.31
 commit : 5f136a7a027bcb54b408a8b16be8767a0b6239de
   date : 2020-03-19
   time : 04-49-37
 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 : 0.1.2
 commit : 957340f6e71d805dbd8b9d04c434b24fd3f92591
   date : 2020-02-16
   time : 06-39-42
 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 : 0.1.3
 commit : 25a6481c509a60ebf5b182f928ded0780dc4ad57
   date : 2020-03-19
   time : 05-16-31
 status : 0
-------------------------------------------------
   name : fc-oogmsh
    tag : 0.2.3
 commit : 4a6082c1f54d867a175cf7e4751769cb8a22844a
   date : 2020-03-19
   time : 04-51-53
 status : 0
-------------------------------------------------
   name : fc-siplt
    tag : 0.2.2
 commit : 2299eabc4604bfb8f6f00d700d54d2531b62cad4
   date : 2020-03-19
   time : 06-01-13
 status : 0
-------------------------------------------------
   name : fc-simesh
    tag : 0.4.2
 commit : e6cb4dd5ff8b00348eddca511f1e37368980fd83
   date : 2020-03-19
   time : 06-13-42
 status : 0
-------------------------------------------------
```

git informations on the LaTeX package used to build this manual

```
-------------------------------------------------
   name : fctools
    tag :
 commit : 57968c4a96c2593cccc9da9efd3e52b2ff012cb5
   date : 2020-02-07
   time : 06:41:09
 status : 1
-------------------------------------------------
```

Using the remote configuration repository:

url	`ssh://lagagit/MCS/Cuvelier/Matlab/fc-config`
commit	`ca2a4f11eb918d3020f934e3545abef8b49ef3e8`