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

François Cuvelier[2]

2017/06/14

[1] Compiled with Python 3.6.0

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

This work was partially supported by ANR Dedales.

Abstract

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

The two FC-SIMESH add-ons FC-SIMESH-MATPLOTLIB [2] and FC-SIMESH-MAYAVI [3] allows a great flexibility in graphical representations of the meshes and datas on the meshes by using respectively the MATPLOTLIB and the MAYAVI packages.

The FC-VFEM$\mathbb{P}_1$ package also contains the techniques of vectorization presented in [5] and extended in [4] and allows good performances when using $\mathbb{P}^1$-Lagrange finite element method.

Contents

Chapter 1

Generic Boundary Value Problems

The notations of [8] are employed in this section and extended to the vector case.

1.1 Scalar boundary value problem

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

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

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

where $\mathbb{A} \in (L^\infty(\Omega))^{d\times d}$, $\boldsymbol{b} \in (L^\infty(\Omega))^d$, $\boldsymbol{c} \in (L^\infty(\Omega))^d$ and $a_0 \in L^\infty(\Omega)$ are given functions and $\langle\cdot,\cdot\rangle$ is the usual scalar product in $\mathbb{R}^d$. We use the same notations as in the chapter 6 of [8] and we note that we can omit either $\operatorname{div}(\boldsymbol{b}u)$ or $\langle \boldsymbol{\nabla} u, \boldsymbol{c}\rangle$ if $\boldsymbol{b}$ and $\boldsymbol{c}$ are sufficiently regular functions. We keep both terms with $\boldsymbol{b}$ and $\boldsymbol{c}$ to deal with more boundary conditions. It should be also noted that it is important to preserve the two terms $\boldsymbol{b}$ and $\boldsymbol{c}$ in the generic formulation to enable a greater flexibility in the choice of the boundary conditions.

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

A *scalar* boundary value problem is given by

Scalar BVP 1 : generic problem

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

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

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

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

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

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

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

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

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

Usual BVP 1 : 2D condenser problem

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

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

where Ω and its boundaries are given in Figure 1.1.

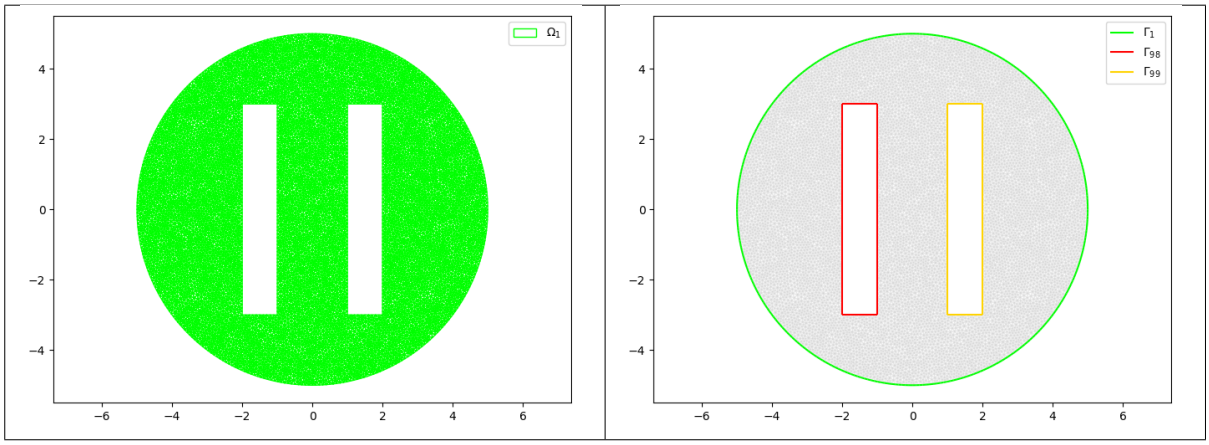


Figure 1.1: 2D condenser mesh and boundaries (left) and numerical solution (right)

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

Scalar BVP 2 : 2D condenser problem

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

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

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

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

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

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

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

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

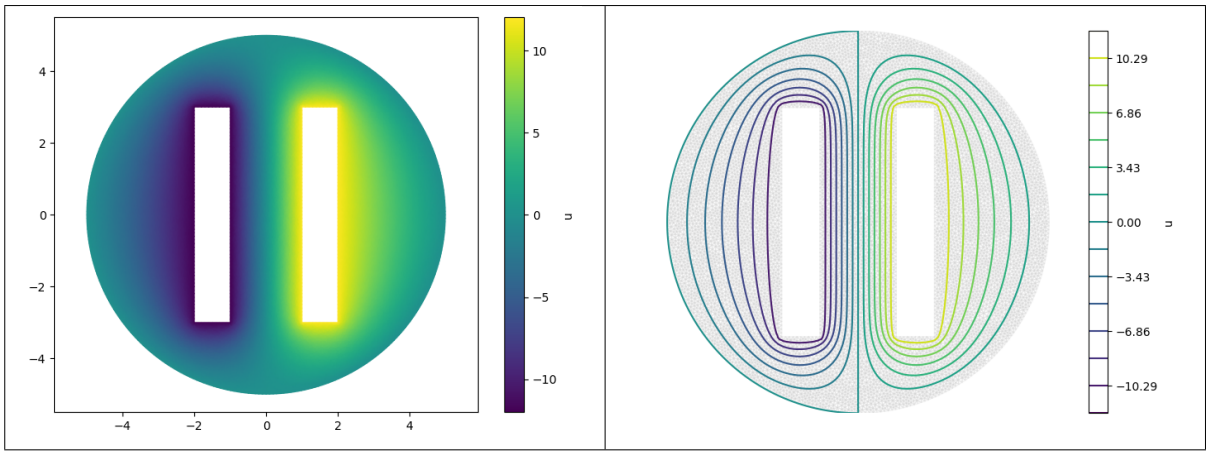


Figure 1.2: 2D condenser numerical solution

1.2 Vector boundary value problem

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

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

where

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

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

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

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

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

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

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

A *vector* boundary value problem is given by

Vector BVP 1 : generic problem

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

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

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

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

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

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

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

Usual vector BVP 1 : 2D simple vector problem

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

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

where Ω and its boundaries are given in Figure 1.1.

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

Vector BVP 2 : 2D simple vector problem

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

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

where

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

$$f \equiv 0,$$

and

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

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

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

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

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

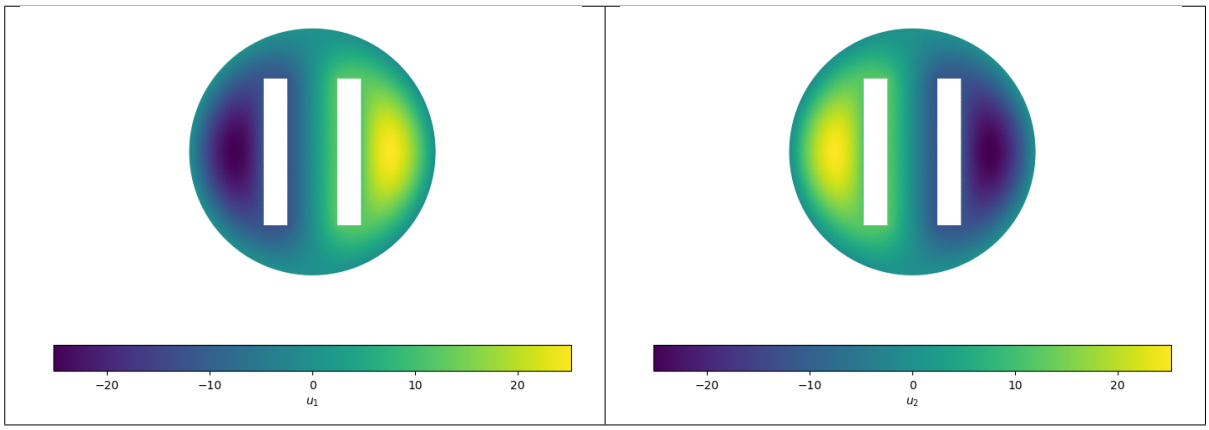


Figure 1.3: Funny vector BVP, u_1 numerical solution (left) and u_2 numerical solution (right)

Chapter 2

Biharmonic Boundary Value Problems

Let $\Omega \subset \mathbb{R}^{\dim}$ and $\Gamma = \partial\Omega$. The biharmonic equation is the fourth-order partial PDE given by

$$\Delta^2 u = f, \quad \text{in } \Omega \tag{2.1}$$

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

The boundary conditions on Γ can be

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

$$u = \frac{\partial u}{\partial \boldsymbol{n}} = g \tag{2.2}$$

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

$$u = \Delta u = g \tag{2.3}$$

- *Pure Hinged Plate* (PHP) or *Steklov* type :

$$u = \Delta u - (1-\sigma)K\frac{\partial u}{\partial \boldsymbol{n}} = g \tag{2.4}$$

- *Cahn-Hilliard* (CH) type

$$\frac{\partial u}{\partial n} = \frac{\partial \Delta u}{\partial n} = g \tag{2.5}$$

2.1 Link with $\mathcal{H}$-operator

Classically the fourth-order PDE (2.1) is converted to the two second-order PDE

$$-\Delta u = v \tag{2.6}$$
$$-\Delta v = f \tag{2.7}$$

These two equations can be equivalently written as

$$\mathcal{G}\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} f \\ 0 \end{pmatrix} \quad \text{or} \quad \mathcal{K}\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} 0 \\ f \end{pmatrix} \tag{2.8}$$

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

$$\mathcal{G} = \begin{pmatrix} 0 & \mathcal{L}_{\mathbb{I},\mathbf{0},\mathbf{0},0} \\ \mathcal{L}_{\mathbb{I},\mathbf{0},\mathbf{0},0} & \mathcal{L}_{\mathbb{O},\mathbf{0},\mathbf{0},-1} \end{pmatrix} \quad \text{and} \quad \mathcal{K} = \begin{pmatrix} \mathcal{L}_{\mathbb{I},\mathbf{0},\mathbf{0},0} & \mathcal{L}_{\mathbb{O},\mathbf{0},\mathbf{0},-1} \\ \mathcal{L}_{\mathbb{O},\mathbf{0},\mathbf{0},0} & \mathcal{L}_{\mathbb{I},\mathbf{0},\mathbf{0},0} \end{pmatrix} \tag{2.9}$$

The Python code using the FC-VFEM$\mathbb{P}_1$ package to create the operators $\mathcal{G}$ and $\mathcal{K}$ are respectively given in Listings 2.1 and 2.2.

```
from fc_vfemp1.operators import
    Loperator,Hoperator
Lop=Loperator(dim=2,
    A=[[1,None],[None,1]])
Gop=Hoperator(dim=2,m=2)
Gop.H[0][1]=Gop.H[1][0]=Lop
Gop.H[1][1]=Loperator(dim=2,a0=-1)
```

Listing 2.1: $\mathcal{G}$ operator with the FC-VFEM$\mathbb{P}_1$ packagein 2D

```
from fc_vfemp1.operators import
    Loperator,Hoperator
Lop=Loperator(dim=2,
    A=[[1,None],[None,1]])
Kop=Hoperator(dim=2,m=2)
Kop.H[0][0]=Kop.H[1][1]=Lop
Kop.H[0][1]=Loperator(dim=2,a0=-1)
```

Listing 2.2: $\mathcal{K}$ operator with the FC-VFEM$\mathbb{P}_1$ package in 2D

Let $\boldsymbol{w} = (u, v)$. With the operator $\mathcal{K}$ given in (2.8), the components of the conormal derivative of $\boldsymbol{w}$ defined in (1.17) are given by

$$\begin{aligned} \frac{\partial \boldsymbol{w}}{\partial n_{\mathcal{K}_1}} &\overset{\text{def}}{=} \sum_{\beta=1}^{2} \frac{\partial \boldsymbol{w}_\beta}{\partial n_{\mathcal{K}_{1,\beta}}} = \sum_{\beta=1}^{2} \left\langle \mathbb{A}^{1,\beta} \boldsymbol{\nabla} \boldsymbol{w}_\beta, \boldsymbol{n} \right\rangle - \left\langle \boldsymbol{b}^{1,\beta} \boldsymbol{u}_\beta, \boldsymbol{n} \right\rangle \\ &= \left\langle \mathbb{I} \boldsymbol{\nabla} \boldsymbol{w}_1, \boldsymbol{n} \right\rangle = \left\langle \boldsymbol{\nabla} u, \boldsymbol{n} \right\rangle \\ &= \frac{\partial u}{\partial \boldsymbol{n}} \end{aligned} \tag{2.10}$$

and

$$\begin{aligned} \frac{\partial \boldsymbol{w}}{\partial n_{\mathcal{K}_2}} &\overset{\text{def}}{=} \sum_{\beta=1}^{2} \frac{\partial \boldsymbol{w}_\beta}{\partial n_{\mathcal{K}_{2,\beta}}} = \sum_{\beta=1}^{2} \left\langle \mathbb{A}^{2,\beta} \boldsymbol{\nabla} \boldsymbol{w}_\beta, \boldsymbol{n} \right\rangle - \left\langle \boldsymbol{b}^{2,\beta} \boldsymbol{u}_\beta, \boldsymbol{n} \right\rangle \\ &= \left\langle \mathbb{I} \boldsymbol{\nabla} \boldsymbol{w}_2, \boldsymbol{n} \right\rangle = \left\langle \boldsymbol{\nabla} v, \boldsymbol{n} \right\rangle \\ &= \frac{\partial v}{\partial \boldsymbol{n}} \end{aligned} \tag{2.11}$$

So with $\mathcal{K}$ operator one can impose the following boundary conditions

$$\begin{aligned} \boldsymbol{w}_\alpha &= g_\alpha^D && \text{on } \Gamma_\alpha^D, \ \forall \alpha \in [\![1,2]\!], \\ \frac{\partial \boldsymbol{w}}{\partial n_{\mathcal{K}_\alpha}} + a_\alpha^R \boldsymbol{w}_\alpha &= g_\alpha^R && \text{on } \Gamma_\alpha^R, \ \forall \alpha \in [\![1,2]\!] \end{aligned}$$

i.e.

$$\begin{aligned} u &= g_1^D && \text{on } \Gamma_1^D, & v &= g_2^D && \text{on } \Gamma_2^D, \\ \frac{\partial u}{\partial \boldsymbol{n}} + a_1^R u &= g_1^R && \text{on } \Gamma_1^R, & \frac{\partial v}{\partial \boldsymbol{n}} + a_2^R v &= g_2^R && \text{on } \Gamma_2^R \end{aligned}$$

Remark 2.1 One can neither impose *clamped plate* (2.2) nor *Pure Hinged Plate* (2.4) boundary conditions with $\mathcal{K}$ operator. This is why thereafter we will only use the $\mathcal{G}$ operator.

In the same way, with the operator $\mathcal{G}$ given in (2.8), the components of the conormal derivative of $\boldsymbol{w}$ defined in (1.17) are given by

$$\begin{aligned} \frac{\partial \boldsymbol{w}}{\partial n_{\mathcal{G}_1}} &\overset{\text{def}}{=} \sum_{\beta=1}^{2} \frac{\partial \boldsymbol{w}_\beta}{\partial n_{\mathcal{G}_{1,\beta}}} = \sum_{\beta=1}^{2} \left\langle \mathbb{A}^{1,\beta} \boldsymbol{\nabla} \boldsymbol{w}_\beta, \boldsymbol{n} \right\rangle - \left\langle \boldsymbol{b}^{1,\beta} \boldsymbol{u}_\beta, \boldsymbol{n} \right\rangle \\ &= \left\langle \mathbb{I} \boldsymbol{\nabla} \boldsymbol{w}_2, \boldsymbol{n} \right\rangle = \left\langle \boldsymbol{\nabla} v, \boldsymbol{n} \right\rangle \\ &= \frac{\partial v}{\partial \boldsymbol{n}} \end{aligned} \tag{2.12}$$

and

$$\begin{aligned}\frac{\partial \boldsymbol{w}}{\partial n_{\mathcal{G}_2}} &\overset{\text{def}}{=} \sum_{\beta=1}^{2} \frac{\partial \boldsymbol{w}_\beta}{\partial n_{\mathcal{G}_{2,\beta}}} = \sum_{\beta=1}^{2} \left\langle \mathbb{A}^{2,\beta} \boldsymbol{\nabla} \boldsymbol{w}_\beta, \boldsymbol{n} \right\rangle - \left\langle \boldsymbol{b}^{2,\beta} \boldsymbol{u}_\beta, \boldsymbol{n} \right\rangle \\ &= \left\langle \mathbb{I} \boldsymbol{\nabla} \boldsymbol{w}_1, \boldsymbol{n} \right\rangle = \left\langle \boldsymbol{\nabla} u, \boldsymbol{n} \right\rangle \\ &= \frac{\partial u}{\partial \boldsymbol{n}} \end{aligned} \tag{2.13}$$

Let us denotes the two partitions avec the boundary Γ:

$$\mathring{\Gamma}_\alpha^D \cap \mathring{\Gamma}_\alpha^R = \varnothing \quad \text{and} \quad \Gamma_\alpha^D \cup \Gamma_\alpha^R = \Gamma, \quad \forall \alpha \in [\![1,2]\!] \tag{2.14}$$

From *Vector* BVP (1.14)-(1.16), using $\mathcal{G}$ operator one can impose the following boundary conditions

$$\begin{aligned} \boldsymbol{w}_\alpha &= g_\alpha^D && \text{on } \Gamma_\alpha^D, \ \forall \alpha \in [\![1,2]\!], \\ \frac{\partial \boldsymbol{w}}{\partial n_{\mathcal{G}_\alpha}} + a_\alpha^R \boldsymbol{w}_\alpha &= g_\alpha^R && \text{on } \Gamma_\alpha^R, \ \forall \alpha \in [\![1,2]\!] \end{aligned}$$

i.e.

$$\begin{aligned} u &= g_1^D && \text{on } \Gamma_1^D, & v &= g_2^D && \text{on } \Gamma_2^D, \\ \frac{\partial v}{\partial \boldsymbol{n}} + a_1^R u &= g_1^R && \text{on } \Gamma_1^R, & \frac{\partial u}{\partial \boldsymbol{n}} + a_2^R v &= g_2^R && \text{on } \Gamma_2^R \end{aligned}$$

To resume, we give a generic biharmonic BVP using the operator $\mathcal{G}$ in *Vector* BVP (3) which is equivalent to the generic mixed formulation for the biharmonic BVP given in *Vector* BVP (4)

Vector BVP 3 : generic biharmonic BVP with $\mathcal{G}$ operator

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

$$\mathcal{G}(\boldsymbol{w}) = \begin{pmatrix} f \\ 0 \end{pmatrix} \quad \text{in } \Omega, \tag{2.15}$$

$$\begin{aligned} \boldsymbol{w}_1 &= g_1^D && \text{on } \Gamma_1^D, & \boldsymbol{w}_2 &= g_2^D && \text{on } \Gamma_2^D, \end{aligned} \tag{2.16}$$

$$\begin{aligned} \frac{\partial \boldsymbol{w}}{\partial n_{\mathcal{G}_1}} + a_1^R \boldsymbol{w}_1 &= g_1^R && \text{on } \Gamma_1^R, & \frac{\partial \boldsymbol{w}}{\partial n_{\mathcal{G}_2}} + a_2^R \boldsymbol{w}_2 &= g_2^R && \text{on } \Gamma_2^R. \end{aligned} \tag{2.17}$$

Vector BVP 4 : generic mixed formulation for the biharmonic BVP

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

$$\begin{pmatrix} 0 & \mathcal{L}_{\mathbb{I},\mathbf{0},\mathbf{0},0} \\ \mathcal{L}_{\mathbb{I},\mathbf{0},\mathbf{0},0} & \mathcal{L}_{\mathbb{O},\mathbf{0},\mathbf{0},-1} \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} f \\ 0 \end{pmatrix} \quad \text{in } \Omega, \tag{2.18}$$

$$\begin{aligned} u &= g_1^D && \text{on } \Gamma_1^D, & v &= g_2^D && \text{on } \Gamma_2^D, \end{aligned} \tag{2.19}$$

$$\begin{aligned} \frac{\partial v}{\partial \boldsymbol{n}} + a_1^R u &= g_1^R && \text{on } \Gamma_1^R, & \frac{\partial u}{\partial \boldsymbol{n}} + a_2^R v &= g_2^R && \text{on } \Gamma_2^R. \end{aligned} \tag{2.20}$$

It's very easy to write (2.15) (or (2.18)) from the generic formulation of the biharmonic BVP with $\mathcal{G}$ operator with the FC-VFEM$\mathbb{P}_1$ package: the source code is given in Listing 2.4 where `Th` is a given `siMesh` object and `f` a given python function or scalar.

```
from fc_vfemp1.operators import Loperator,Hoperator
from fc_vfemp1.BVP import BVP,PDE
Lop=Loperator(dim=2,A=[[1,None],[None,1]])
Gop=Hoperator(dim=2,m=2)
Gop.H[0][1]=Gop.H[1][0]=Lop
Gop.H[1][1]=Loperator(dim=2,a0=-1)
pde=PDE(Op=Gop, f=[f,0])
bvp=BVP(Th, pde=pde)
```

Listing 2.3: Writing the $\mathcal{G}(\boldsymbol{w}) = \begin{pmatrix} f \\ 0 \end{pmatrix}$ with the FC-VFEM$\mathbb{P}_1$ package

or more concisely FC-VFEM$\mathbb{P}_1$-BIHARMONIC package:

```
import fc_vfemp1_biharmonic.lib as blib
bvp=blib.BiharmonicBVP(Th, f=f)
```

Listing 2.4: Writing shortly the $\mathcal{G}(\boldsymbol{w}) = (f, 0)^t$ with the FC-VFEM$\mathbb{P}_1$-BIHARMONIC package

Remark 2.2 By default the boundary conditions of a `BVP` object are set to **homogeneous Neumann** so we have

$$\frac{\partial \boldsymbol{w}}{\partial n_{\mathcal{G}_1}} = \frac{\partial \boldsymbol{w}}{\partial n_{\mathcal{G}_2}} = 0 \quad \Leftrightarrow \quad \frac{\partial v}{\partial \boldsymbol{n}} = \frac{\partial u}{\partial \boldsymbol{n}} = 0$$

which is the homogeneous **Cahn-Hilliard** boundary condition.

Now we will see in the next sections how to set the **Clamped Plate** (CP), **Simply Supported Plate** (CP) and **Cahn-Hilliard** (CH) boundary conditions.

2.2 Some boundary conditions

Let $\Gamma_{\text{lab}} \subset \Gamma$. We want to set one of the following boundary conditions on Γ_{lab}:

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

$$u = g_a \quad \textbf{and} \quad \frac{\partial u}{\partial \boldsymbol{n}} = g_b \quad \text{on } \Gamma_{\text{lab}} \tag{2.21}$$

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

$$u = g_a \quad \textbf{and} \quad \Delta u = -g_b \quad \text{on } \Gamma_{\text{lab}} \tag{2.22}$$

- *Cahn-Hilliard* (CH) type

$$\frac{\partial u}{\partial n} = g_a \quad \textbf{and} \quad \frac{\partial \Delta u}{\partial n} = -g_b \quad \text{on } \Gamma_{\text{lab}} \tag{2.23}$$

Now we will see how to rewrite theses boundary conditions as those in *Vector* BVP 4, equations (2.19)-(2.20), and *Vector* BVP 3, equations (2.16)-(2.17).

2.2.1 Clamped Plate boundary condition

From (2.19) and (2.20), we deduce that (2.21) imposes

$$u = g_a \quad \text{on } \Gamma_{\text{lab}} \subset \Gamma_1^D \quad \text{and} \quad \frac{\partial u}{\partial \boldsymbol{n}} = g_b \quad \text{on } \Gamma_{\text{lab}} \subset \Gamma_2^R$$

So with $\mathcal{G}$ operator and with $\boldsymbol{w} = (u, v)$ we obtain

$$w_1 = g_a \quad \text{on } \Gamma_{\text{lab}} \subset \Gamma_1^D \quad \text{and} \quad \frac{\partial \boldsymbol{w}}{\partial n_{\mathcal{G}_2}} = g_b \quad \text{on } \Gamma_{\text{lab}} \subset \Gamma_2^R$$

Let `bvp` be the `BVP` object build in Listing 2.4. We want to set this object with the (CP) boundary condition. To set the dirichlet condition on first component $w_1 = g_a$ on Γ_{lab}, we can use the `setDirichlet` method of the `BVP` object:

```
bvp.setDirichlet(lab,ga,comps=[0])
```

As python uses 0-based indexing, the first component is selected with `comps=[0]` .
To set the Neumann condition on second component $\frac{\partial \boldsymbol{w}}{\partial n_{\mathcal{G}_2}} = g_b$ we can use the `setRobin` method of the `BVP` object:

```
bvp.setRobin(lab,gb,comps=[1])
```

As python uses 0-based indexing, the second component is selected with `comps=[1]` .

A more convenient way is to use the dedicated function `setClampedPlate` function:

```
import fc_vfemp1_biharmonic.lib as blib
blib.setClampedPlate(bvp,lab,[ga,gb])
```

Listing 2.5: Setting a (CP) boundary condition

2.2.2 Simply Supported Plate boundary condition

From (2.19) and (2.20), we deduce that (2.22) imposes

$$u = g_a \text{ on } \Gamma_{\text{lab}} \subset \Gamma_1^D \quad \text{and} \quad v = g_b \text{ on } \Gamma_{\text{lab}} \subset \Gamma_2^D$$

So with $\mathcal{G}$ operator and with $\boldsymbol{w} = (u, v)$ we obtain

$$w_1 = g_a \text{ on } \Gamma_{\text{lab}} \subset \Gamma_1^D \quad \text{and} \quad w_2 = g_b \text{ on } \Gamma_{\text{lab}} \subset \Gamma_2^D$$

Let `bvp` be the `BVP` object build in Listing 2.4. We want to set this object with the (SSP) boundary condition. To set the dirichlet conditions, we can use the `setDirichlet` method of the `BVP` object:

```
bvp.setDirichlet(lab,ga,comps=[0])
bvp.setDirichlet(lab,gb,comps=[1])
```

or

```
bvp.setDirichlet(lab,[ga,gb])
```

A more convenient way is to use the dedicated function `setSimplySupportedPlate` function:

```
import fc_vfemp1_biharmonic.lib as blib
blib.setSimplySupportedPlate(bvp,lab,[ga,gb])
```

Listing 2.6: Setting a (SSP) boundary condition

2.2.3 Cahn-Hilliard boundary condition

From (2.19) and (2.20), we deduce that (2.23) imposes

$$\frac{\partial v}{\partial n} = g_b \text{ on } \Gamma_{\text{lab}} \subset \Gamma_1^R \quad \text{and} \quad \frac{\partial u}{\partial \boldsymbol{n}} = g_a \text{ on } \Gamma_{\text{lab}} \subset \Gamma_2^R$$

where $v = -\Delta u$.

So with $\mathcal{G}$ operator and with $\boldsymbol{w} = (u, v)$ we obtain

$$\frac{\partial \boldsymbol{w}}{\partial n_{\mathcal{G}_1}} = g_b \text{ on } \Gamma_{\text{lab}} \subset \Gamma_1^R \quad \text{and} \quad \frac{\partial \boldsymbol{w}}{\partial n_{\mathcal{G}_2}} = g_a \text{ on } \Gamma_{\text{lab}} \subset \Gamma_2^R$$

Let `bvp` be the `BVP` object build in Listing 2.4. We want to set this object with the (CH) boundary condition. To set the Neumann conditions, we can use the `setRobin` method of the `BVP` object:

```
bvp.setRobin(lab,gb,comps=[0])
bvp.setRobin(lab,ga,comps=[1])
```

or more concisely

`bvp.setRobin(lab,[gb,ga])`

A more convenient way is to use the dedicated function `setCahnHilliard` function:

```
import fc_vfemp1_biharmonic.lib as blib
blib.setCahnHilliard(bvp,lab,[ga,gb])
```

Listing 2.7: Setting a (CH) boundary condition

2.3 Examples with exact solutions

To obtain biharmonic problems with exact solutions we give use the exact solution and using **sympy** package we compute the functions $v = -\Delta u_{\text{ex}}$, $f = -\Delta v$, $\frac{\partial u_{\text{ex}}}{\partial x}$, $\frac{\partial u_{\text{ex}}}{\partial y}$, $\frac{\partial v}{\partial x}$ and $\frac{\partial v}{\partial y}$.

Then on simple geometry (square,disk, ...) the RHS terms in the boundary conditions (i.e. g_a or g_b) can be analytically computed since the exterior normals are very simple .

Usual BVP 2 : Biharmonic problem with _clamped plate_ (CP) boundary conditions

Find u such that

$$\begin{aligned} &\Delta^2 u = f, && \text{in } \Omega \subset \mathbb{R}^2 \\ &u = g_a \quad \textbf{and} \quad \tfrac{\partial u}{\partial n} = g_b && \text{on } \Gamma \end{aligned}$$

Usual BVP 3 : Biharmonic problem with _simply supported plate_ (SSP) boundary conditions

Find u such that

$$\begin{aligned} &\Delta^2 u = f, && \text{in } \Omega \subset \mathbb{R}^2 \\ &u = g_a \quad \textbf{and} \quad \Delta u = g_b && \text{on } \Gamma \end{aligned}$$

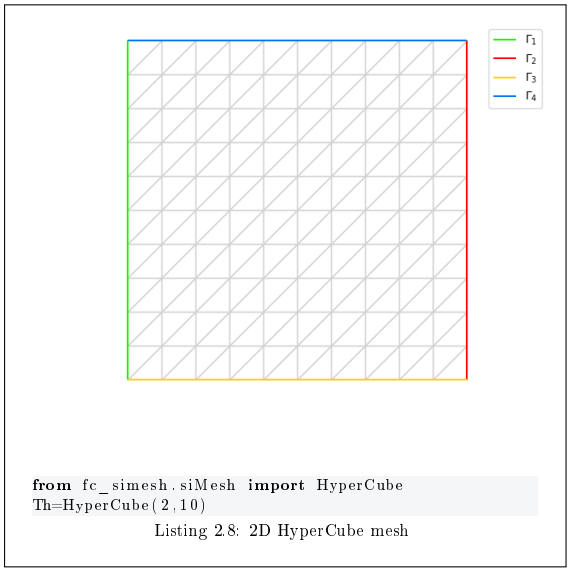
Usual BVP 4 : Biharmonic problem with _Cahn-Hilliard_ (CH) boundary conditions

Find u such that

$$\begin{aligned} &\Delta^2 u = f, && \text{in } \Omega \subset \mathbb{R}^2 \\ &\tfrac{\partial u}{\partial n} = g_a \quad \textbf{and} \quad \tfrac{\partial \Delta u}{\partial n} = g_b && \text{on } \Gamma \end{aligned}$$

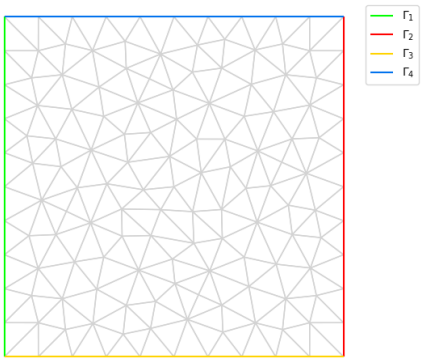
2.3.1 Clamped plate problems on the unit square

The mesh of the unit square $[0,1] \times [0,1]$ can be obtained by using the `HyperCube` function of the FC-SIMESH package and we obtain a regular mesh. An other way is to generate a mesh file by using `gmsh` and a given *.geo* file. This can be done entirely by using FC-OOGMSH and FC-SIMESH packages, and the *square4.geo* file. The complete codes are given in Listing 3 and 10 with graphical representation of the obtained meshes. One can see that the label of the boundaries are the same on both meshes.




```
from fc_simesh.siMesh import HyperCube
Th=HyperCube(2,10)
```

Listing 2.8: 2D HyperCube mesh




```
import os
from fc_oogmsh import gmsh
from fc_simesh.siMesh import siMesh
from fc_vfemp1_biharmonic.sys import get_geo
geofile='square4'
(geodir,geofile)=get_geo(2,2,geofile)
geoFile=geodir+os.sep+geofile+'.geo'
meshfile=gmsh.buildmesh2d(geoFile,10)
Th=siMesh(meshfile)
```

Listing 2.9: 2D mesh of square with `gmsh`

In file `biharmonic_CP_square_ex.py`[1]) the `run` function can solve the clamped plate problem, *Usual* BVP 2, on the unit square for a given *exact* solution u_{ex} as a string (default is `'sin(x)*sin(y)'`).

Now we will set the clamped plate boundary conditions on this problem. On each boundary we can compute $g_a = u_{\text{ex}}$ and $g_b = \frac{\partial u_{\text{ex}}}{\partial n} \overset{\text{def}}{=} \langle \boldsymbol{\nabla} u_{\text{ex}}, \boldsymbol{n} \rangle$ where $\boldsymbol{n}$ is the exterior normal to Γ:

$$
\begin{aligned}
g_b = \frac{\partial u_{\text{ex}}}{\partial n} &= \left\langle \boldsymbol{\nabla} u_{\text{ex}}, \begin{pmatrix} -1 \\ 0 \end{pmatrix} \right\rangle = -\frac{\partial u_{\text{ex}}}{\partial x} && \text{on } \Gamma_1 \\
g_b = \frac{\partial u_{\text{ex}}}{\partial n} &= \left\langle \boldsymbol{\nabla} u_{\text{ex}}, \begin{pmatrix} 1 \\ 0 \end{pmatrix} \right\rangle = \frac{\partial u_{\text{ex}}}{\partial x} && \text{on } \Gamma_2 \\
g_b = \frac{\partial u_{\text{ex}}}{\partial n} &= \left\langle \boldsymbol{\nabla} u_{\text{ex}}, \begin{pmatrix} 0 \\ -1 \end{pmatrix} \right\rangle = -\frac{\partial u_{\text{ex}}}{\partial y} && \text{on } \Gamma_3 \\
g_b = \frac{\partial u_{\text{ex}}}{\partial n} &= \left\langle \boldsymbol{\nabla} u_{\text{ex}}, \begin{pmatrix} 0 \\ 1 \end{pmatrix} \right\rangle = \frac{\partial u_{\text{ex}}}{\partial y} && \text{on } \Gamma_4
\end{aligned}
$$

Part of the source code which sets the biharmonic problem with the 4 boundary conditions is given in Listing 2.10

```
    print('*** Setting the BVP')
bvp=blib.BiharmonicBVP(Th, f=f)
blib.setClampedPlate(bvp,1,[uex,lambda x,y: -dudx(x,y)])
blib.setClampedPlate(bvp,2,[uex,lambda x,y:  dudx(x,y)])
blib.setClampedPlate(bvp,3,[uex,lambda x,y: -dudy(x,y)])
blib.setClampedPlate(bvp,4,[uex,lambda x,y:  dudy(x,y)])
```

Listing 2.10: Biharmonic BVP with Clamped Plate (CP) boundary conditions

In Listing 2.11, we solve the clamped plate problem with exact solution $u_{\text{ex}} = \sin(5x+1)\cos(3y^2-1)$ and the output is presented. The graphical results are given in Figure 2.1

[1] directory `fc_vfemp1_biharmonic/examples/`

Listing 2.11: Running biharmonic clamped plate problem with `sol='sin(5*x+1)*cos(3*y**2-1)'`

```
from fc_vfemp1_biharmonic.examples.biharmonic_CP_square_ex import run
res=run(N=200,sol='sin(5*x+1)*cos(3*y**2-1)')
```

Output

```
------------------------------------------------------------
Solving biharmonic Boundary Value Problems (BVP's) with
        fc_vfemp1_biharmonic package
Copyright (C) 2017 Cuvelier F.
  (LAGA/CNRS/Paris 13 University)
------------------------------------------------------------
     2D biharmonic BVP on [0,1]x[0,1]
     Mesh generated with HyperCube function.

             Laplacian(laplacian(u)) = f

 with boundary conditions
   * Clamped Plate (CP)
       u=ga and du/dn=gb on [1,2,3,4]
 Exact solution : sin(5*x+1)*cos(3*y**2-1)
------------------------------------------------------------
*** Setting the mesh using HyperCube function [fc_simesh]
    Mesh sizes : nq=40401, nme=80000, h=7.071e-03
*** Setting the BVP
*** Solving the BVP
*** Computing relative errors
  u: errL2=3.7141e-05, errH1=1.1168e-04, errInf=8.3274e-05
  v: errL2=8.1614e-03, errH1=5.1560e-01, errInf=1.4892e+00
*** Plotting solutions
```

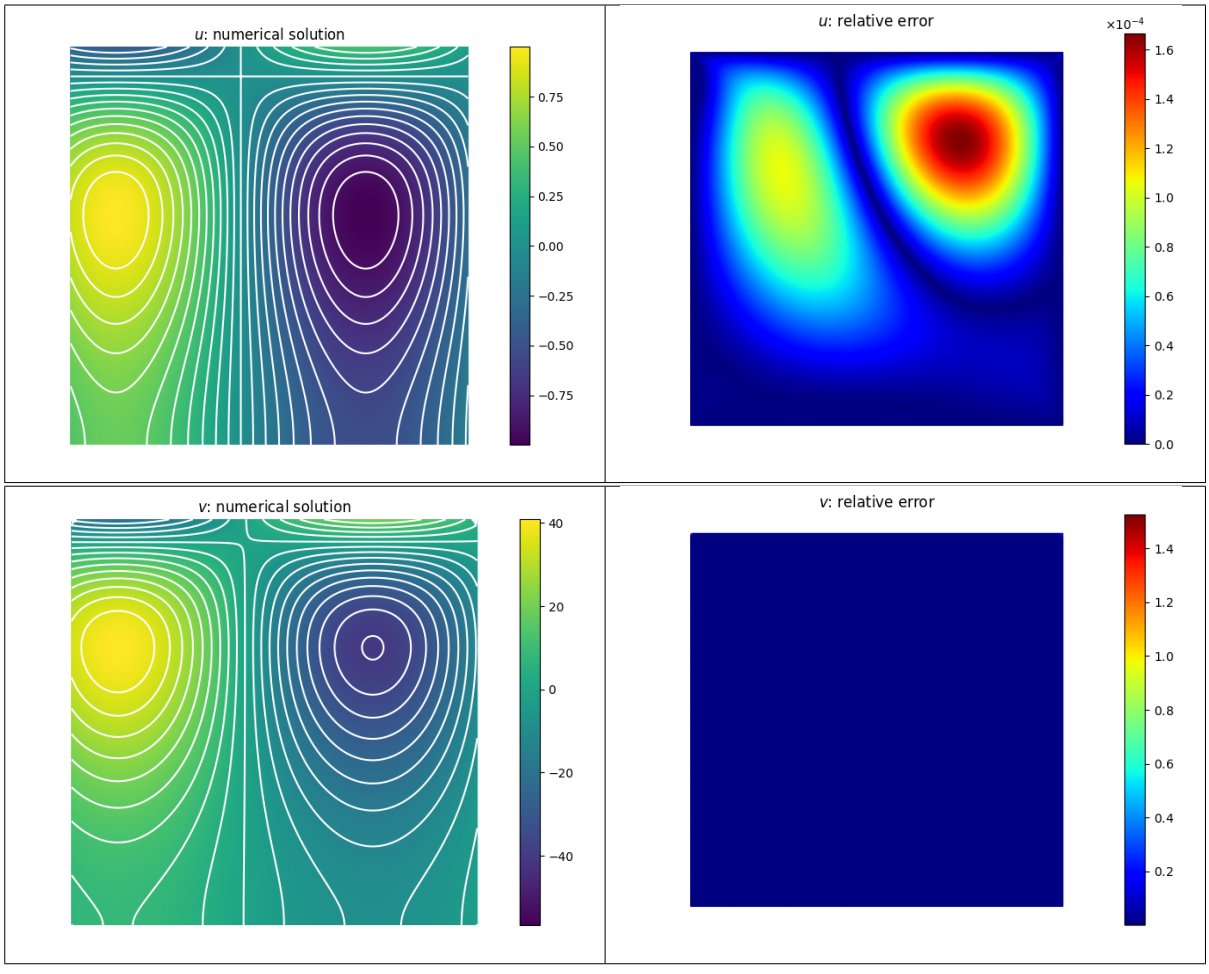


Figure 2.1: 2D biharmonic BVP with CP boundary conditions, u numerical solution (upper left) with error (upper right) and $v = -\Delta u$ numerical solution (bottom left) with error (bottom right)

One can also compute the orders of convergence by using the `order` function. We use this function in Listing 2.12 and Listing 2.13 on respectively the meshes generated with the `HyperCube` function and the meshes given by `gmsh` with `square4.geo` file. Superconvergence phenomenon with the `HyperCube` mesh is illustred in Figure 2.2.

Listing 2.12: Computing orders for the biharmonic clamped plate problem with `sol='sin(5*x+1)*cos(3*y**2-1)'`

```
from fc_vfemp1_biharmonic.examples.biharmonic_CP_square_ex import order
order(LN=[50,100,150,200,250],sol='sin(5*x+1)*cos(3*y**2-1)')
```

Output

```
------------------------------------------------------------
Solving biharmonic Boundary Value Problems (BVP's) with
        fc_vfemp1_biharmonic package
Copyright (C) 2017 Cuvelier F.
  (LAGA/CNRS/Paris 13 University)
------------------------------------------------------------
     2D biharmonic BVP on [0,1]x[0,1]
     Mesh generated with HyperCube function.

             Laplacian(laplacian(u)) = f

 with boundary conditions
   * Clamped Plate (CP)
       u=ga and du/dn=gb on [1,2,3,4]
 Exact solution : sin(5*x+1)*cos(3*y**2-1)
------------------------------------------------------------
[ 1/ 5] N=50
   Mesh sizes : nq= 2601, nme= 5000, h=2.828e-02
   ndof=5202 - Error : uH1=1.80e-03, uL2=5.92e-04, vH1=5.29e-01, vL2=3.27e-02
[ 2/ 5] N=100
   Mesh sizes : nq= 10201, nme= 20000, h=1.414e-02
   ndof=20402 - Error : uH1=4.48e-04, uL2=1.48e-04, vH1=5.22e-01, vL2=1.63e-02
[ 3/ 5] N=150
   Mesh sizes : nq= 22801, nme= 45000, h=9.428e-03
   ndof=45602 - Error : uH1=1.98e-04, uL2=6.60e-05, vH1=5.19e-01, vL2=1.09e-02
[ 4/ 5] N=200
   Mesh sizes : nq= 40401, nme= 80000, h=7.071e-03
   ndof=80802 - Error : uH1=1.12e-04, uL2=3.71e-05, vH1=5.16e-01, vL2=8.16e-03
[ 5/ 5] N=250
   Mesh sizes : nq= 63001, nme= 125000, h=5.657e-03
   ndof=126002 - Error : uH1=7.15e-05, uL2=2.38e-05, vH1=5.14e-01, vL2=6.53e-03
```

Listing 2.13: Computing orders for the biharmonic clamped plate problem with `sol='sin(5*x+1)*cos(3*y**2-1)'`

```
from fc_vfemp1_biharmonic.examples.biharmonic_CP_square_ex import order
order(regular=False,sol='sin(5*x+1)*cos(3*y**2-1)')
```

Output

```
------------------------------------------------------------
Solving biharmonic Boundary Value Problems (BVP's) with
        fc_vfemp1_biharmonic package
Copyright (C) 2017 Cuvelier F.
  (LAGA/CNRS/Paris 13 University)
------------------------------------------------------------
     2D biharmonic BVP on [0,1]x[0,1]
     Mesh generated with gmsh and square4.geo file.

             Laplacian(laplacian(u)) = f

 with boundary conditions
   * Clamped Plate (CP)
       u=ga and du/dn=gb on [1,2,3,4]
 Exact solution : sin(5*x+1)*cos(3*y**2-1)
------------------------------------------------------------
[ 1/ 4] N=25
   Mesh sizes : nq= 891, nme=  1680, h=5.510e-02
   ndof=1782 - Error : uH1=8.54e-03, uL2=1.35e-03, vH1=5.18e-01, vL2=7.43e-02
[ 2/ 4] N=50
   Mesh sizes : nq= 3436, nme= 6670, h=2.947e-02
   ndof=6872 - Error : uH1=3.86e-03, uL2=3.39e-04, vH1=5.13e-01, vL2=3.30e-02
[ 3/ 4] N=100
   Mesh sizes : nq= 13469, nme= 26536, h=1.503e-02
   ndof=26938 - Error : uH1=1.88e-03, uL2=8.90e-05, vH1=7.60e-01, vL2=2.51e-02
[ 4/ 4] N=150
   Mesh sizes : nq= 30081, nme= 59560, h=9.872e-03
   ndof=60162 - Error : uH1=1.28e-03, uL2=5.90e-05, vH1=1.03e+00, vL2=2.39e-02
```

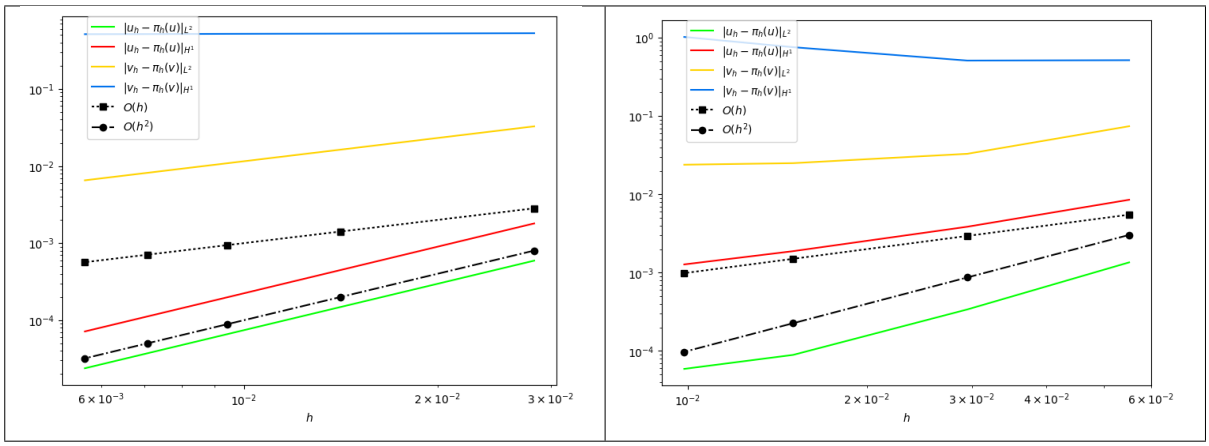
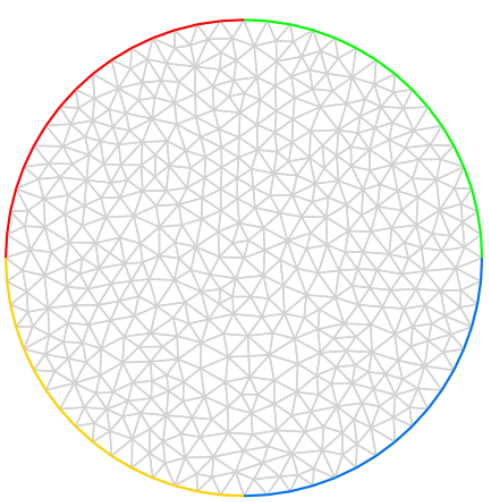


Figure 2.2: 2D biharmonic BVP with CP boundary conditions, order of convergence of u and $v = -\Delta u$ for `HyperCube` mesh (left) and `square4` mesh (right). Superconvergence phenomenon with the `HyperCube` mesh.

2.3.2 Clamped plate problems on the unit disk

This example is very similar to the previous one. So for better understanding it is better to read in advance the section 2.3.1.

The mesh of the unit disk can be obtained by using `gmsh` and a given *.geo* file. This can be done entirely by using FC-OOGMSH and FC-SIMESH packages, and the *disk4bounds.geo* file. The complete code is given in Listing ?? with a graphical representation of the obtained mesh.




```
import os
from fc_oogmsh import gmsh
from fc_simesh.siMesh import siMesh
from fc_vfemp1_biharmonic.sys import get_geo
geofile='disk4bounds'
(geodir,geofile)=get_geo(2,2,geofile)
geoFile=geodir+os.sep+geofile+'.geo'
meshfile=gmsh.buildmesh2d(geoFile,10)
Th=siMesh(meshfile)
```

Listing 2.14: 2D mesh of unit disk with `gmsh`

In file `biharmonic_CP_disk_ex.py`[2]) the `run` function can solve a (generic) clamped plate problem, *Usual* BVP 2, on the unit disk for a given *exact* solution u_{ex} as a string (default is `'sin(x)*sin(y)'`).

[2] directory `fc_vfemp1_biharmonic/examples/`

Now we will set the clamped plate boundary conditions on this problem. On each boundary we can compute $g_a = u_{\text{ex}}$ and $g_b = \frac{\partial u_{\text{ex}}}{\partial n} \overset{\text{def}}{=} \langle \boldsymbol{\nabla} u_{\text{ex}}, \boldsymbol{n} \rangle$ where $\boldsymbol{n} = \begin{pmatrix} x \\ y \end{pmatrix}$ is the exterior normal to Γ:

$$g_b = \frac{\partial u_{\text{ex}}}{\partial n} = \left\langle \boldsymbol{\nabla} u_{\text{ex}}, \begin{pmatrix} x \\ y \end{pmatrix} \right\rangle = x \frac{\partial u_{\text{ex}}}{\partial x} + y \frac{\partial u_{\text{ex}}}{\partial y} \qquad \text{on } \Gamma$$

Part of the source code which sets the biharmonic problem with the 4 (CP) boundary conditions is given in Listing 2.15

```
bvp=blib.BiharmonicBVP(Th, f=f)
ga=uex; gb=lambda x,y: x*dudx(x,y)+y*dudy(x,y)
for lab in [1,2,3,4]:
    blib.setClampedPlate(bvp,lab,[ga,gb])
```

Listing 2.15: Biharmonic BVP with Clamped Plate (CP) boundary conditions on the unit disk

In Listing 2.16, we solve the clamped plate problem with exact solution $u_{\text{ex}} = \sin(5x+1)\cos(3y^2-1)$ and the output is presented. The graphical results are given in Figure 2.3

Listing 2.16: Running biharmonic clamped plate problem with `sol='sin(5*x+1)*cos(3*y**2-1)'` on the unit disk

```
from fc_vfemp1_biharmonic.examples.biharmonic_CP_disk_ex import run
res=run(N=150,sol='sin(5*x+1)*cos(3*y**2-1)')
```

Output

```
-----------------------------------------------------------------
Solving biharmonic Boundary Value Problems (BVP's) with
        fc_vfemp1_biharmonic package
Copyright (C) 2017 Cuvelier F.
  (LAGA/CNRS/Paris 13 University)
-----------------------------------------------------------------
     2D biharmonic BVP on unit disk
     Mesh generated with gmsh and disk4bounds.geo file.

             Laplacian(laplacian(u)) = f

 with boundary conditions
   * Clamped Plate (CP)
      u=ga and du/dn=gb on [1,2,3,4]
 Exact solution : sin(5*x+1)*cos(3*y**2-1)
-----------------------------------------------------------------
*** Setting the mesh using gmsh and disk4bounds.geo file.
    Mesh sizes : nq=94571, nme=188196, h=1.060e-02
*** Setting the BVP
*** Solving the BVP
*** Computing relative errors
  u: errL2=2.9625e-04, errH1=1.4514e-03, errInf=3.2367e-04
  v: errL2=1.8127e-02, errH1=8.6914e-01, errInf=4.5916e-01
*** Plotting solutions
```

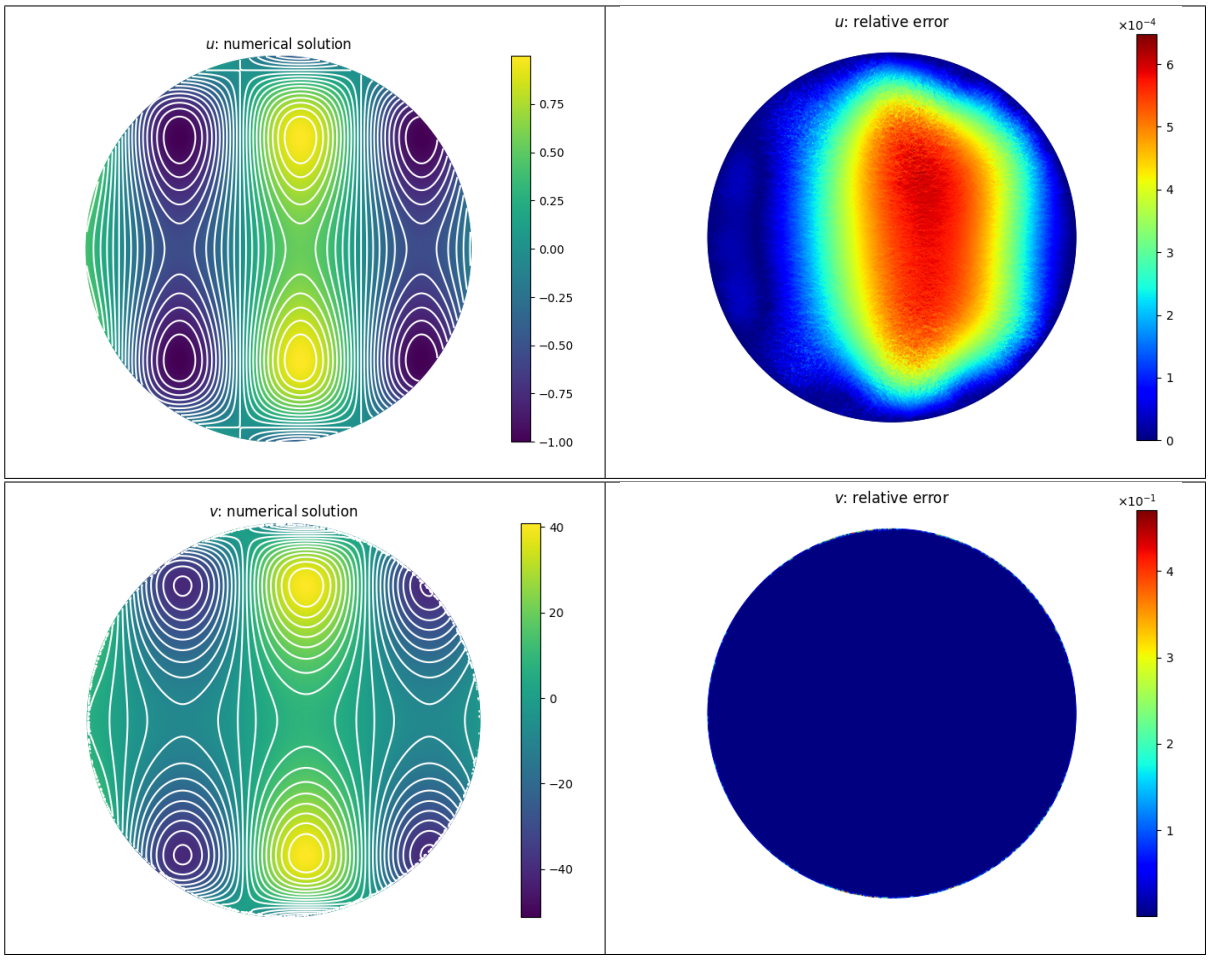


Figure 2.3: 2D biharmonic BVP with CP boundary conditions on the unit disk, u numerical solution (upper left) with error (upper right) and $v = -\Delta u$ numerical solution (bottom left) with error (bottom right)

One can also compute the orders of convergence by using the `order` function. We use this function in Listing 2.17 and the order are represented in Figure 2.4.

Listing 2.17: Computing orders for the biharmonic clamped plate problem with `sol='sin(5*x+1)*cos(3*y**2-1)'` on the unit disk

```
from fc_vfemp1_biharmonic.examples.biharmonic_CP_disk_ex import order
order(sol='sin(5*x+1)*cos(3*y**2-1)')
```

Output

```
------------------------------------------------------------------
Solving biharmonic Boundary Value Problems (BVP's) with
        fc_vfemp1_biharmonic package
Copyright (C) 2017 Cuvelier F.
  (LAGA/CNRS/Paris 13 University)
------------------------------------------------------------------
     2D biharmonic BVP on unit disk
     Mesh generated with gmsh and disk4bounds.geo file.

            Laplacian(laplacian(u)) = f

 with boundary conditions
   * Clamped Plate (CP)
      u=ga and du/dn=gb on [1,2,3,4]
 Exact solution : sin(5*x+1)*cos(3*y**2-1)
------------------------------------------------------------------
[ 1/ 4] N=25
   Mesh sizes : nq= 2766, nme= 5370, h=5.760e-02
   ndof=5532 - Error : uH1=9.62e-03, uL2=4.81e-03, vH1=4.26e-01, vL2=5.54e-02
[ 2/ 4] N=50
   Mesh sizes : nq= 10668, nme= 21018, h=2.924e-02
   ndof=21336 - Error : uH1=4.28e-03, uL2=1.38e-03, vH1=5.14e-01, vL2=3.19e-02
[ 3/ 4] N=100
   Mesh sizes : nq= 42319, nme= 84004, h=1.539e-02
   ndof=84638 - Error : uH1=2.20e-03, uL2=5.88e-04, vH1=7.62e-01, vL2=2.41e-02
[ 4/ 4] N=150
   Mesh sizes : nq= 94571, nme= 188196, h=1.060e-02
   ndof=189142 - Error : uH1=1.45e-03, uL2=2.96e-04, vH1=8.69e-01, vL2=1.81e-02
```

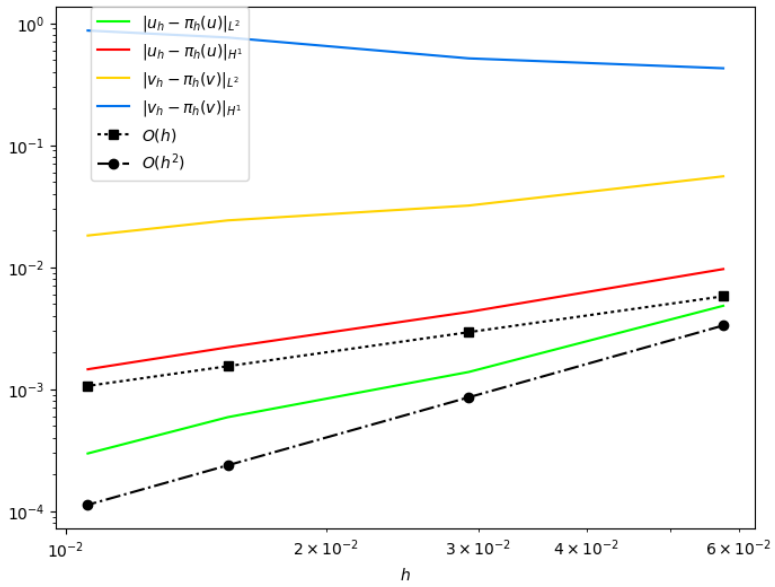


Figure 2.4: 2D biharmonic BVP with CP boundary conditions on the unit disk, order of convergence of u and $v = -\Delta u$.

2.3.3 Simply supported plate problems on the unit square

The mesh of the unit square $[0,1]\times[0,1]$ can be obtained by using the `HyperCube` function of the FC-SIMESH package and we obtain a regular mesh. An other way is to generate a mesh file by using `gmsh` and a given *.geo* file. This can be done entirely by using FC-OOGMSH and FC-SIMESH packages, and the *square4.geo* file. The complete codes are given in Listing 3 and 10 with graphical representation of the obtained meshes. One can see that the label of the boundaries are the same on both meshes.

In file `biharmonic_SSP_square_ex.py`[3]) the `run` function can solve a (generic) clamped plate problem, *Usual* BVP 2, on the unit square for a given *exact* solution u_{ex} as a string (default is `'sin(x)*sin(y)'`).

Now we will set the clamped plate boundary conditions on this problem. On each boundary we can compute $g_a = u_{\text{ex}}$ and $g_b = \frac{\partial u_{\text{ex}}}{\partial n} \overset{\text{def}}{=} \langle \boldsymbol{\nabla} u_{\text{ex}}, \boldsymbol{n}\rangle$ where $\boldsymbol{n}$ is the exterior normal to Γ:

$$
\begin{aligned}
g_b = \frac{\partial u_{\text{ex}}}{\partial n} &= \left\langle \boldsymbol{\nabla} u_{\text{ex}}, \begin{pmatrix} -1 \\ 0 \end{pmatrix} \right\rangle = -\frac{\partial u_{\text{ex}}}{\partial x} && \text{on } \Gamma_1 \\
g_b = \frac{\partial u_{\text{ex}}}{\partial n} &= \left\langle \boldsymbol{\nabla} u_{\text{ex}}, \begin{pmatrix} 1 \\ 0 \end{pmatrix} \right\rangle = \frac{\partial u_{\text{ex}}}{\partial x} && \text{on } \Gamma_2 \\
g_b = \frac{\partial u_{\text{ex}}}{\partial n} &= \left\langle \boldsymbol{\nabla} u_{\text{ex}}, \begin{pmatrix} 0 \\ -1 \end{pmatrix} \right\rangle = -\frac{\partial u_{\text{ex}}}{\partial y} && \text{on } \Gamma_3 \\
g_b = \frac{\partial u_{\text{ex}}}{\partial n} &= \left\langle \boldsymbol{\nabla} u_{\text{ex}}, \begin{pmatrix} 0 \\ 1 \end{pmatrix} \right\rangle = \frac{\partial u_{\text{ex}}}{\partial y} && \text{on } \Gamma_4
\end{aligned}
$$

Part of the source code which sets the biharmonic problem with the 4 (SSP) boundary conditions is given in Listing 2.18

```
bvp=blib.BiharmonicBVP(Th, f=f)
for lab in [1,2,3,4]:
  blib.setSimplySupportedPlate(bvp,lab,[u,v])
```

Listing 2.18: Biharmonic BVP with Simply supported plate (SSP) boundary conditions

In Listing 2.19, we solve the clamped plate problem with exact solution $u_{\text{ex}} = \sin(5x+1)\cos(3y^2-1)$ and the output is presented. The graphical results are given in Figure 2.5

[3] directory `fc_vfemp1_biharmonic/examples/`

Listing 2.19: Running biharmonic clamped plate problem with `sol='sin(5*x+1)*cos(3*y**2-1)'`

```
from fc_vfemp1_biharmonic.examples.biharmonic_SSP_square_ex import run
res=run(N=300,sol='sin(5*x+1)*cos(3*y**2-1)')
```

Output

```
------------------------------------------------------------
Solving biharmonic Boundary Value Problems (BVP's) with
        fc_vfemp1_biharmonic package
Copyright (C) 2017 Cuvelier F.
  (LAGA/CNRS/Paris 13 University)
------------------------------------------------------------
     2D biharmonic BVP on [0,1]x[0,1]
     Mesh generated with HyperCube function.

             Laplacian(laplacian(u)) = f

 with boundary conditions
   * Simply Supported Plate (SSP)
       u=gc and Laplacian(u)=gd on [1,2,3,4]
 Exact solution : sin(5*x+1)*cos(3*y**2-1)
------------------------------------------------------------
*** Setting the mesh using HyperCube function [fc_simesh]
    Mesh sizes : nq=90601, nme=180000, h=4.714e-03
*** Setting the BVP
*** Solving the BVP
*** Computing relative errors
  u: errL2=2.4130e-05, errH1=6.9101e-05, errInf=4.9492e-05
  v: errL2=4.9773e-05, errH1=6.4835e-05, errInf=6.7332e-05
*** Plotting solutions
```

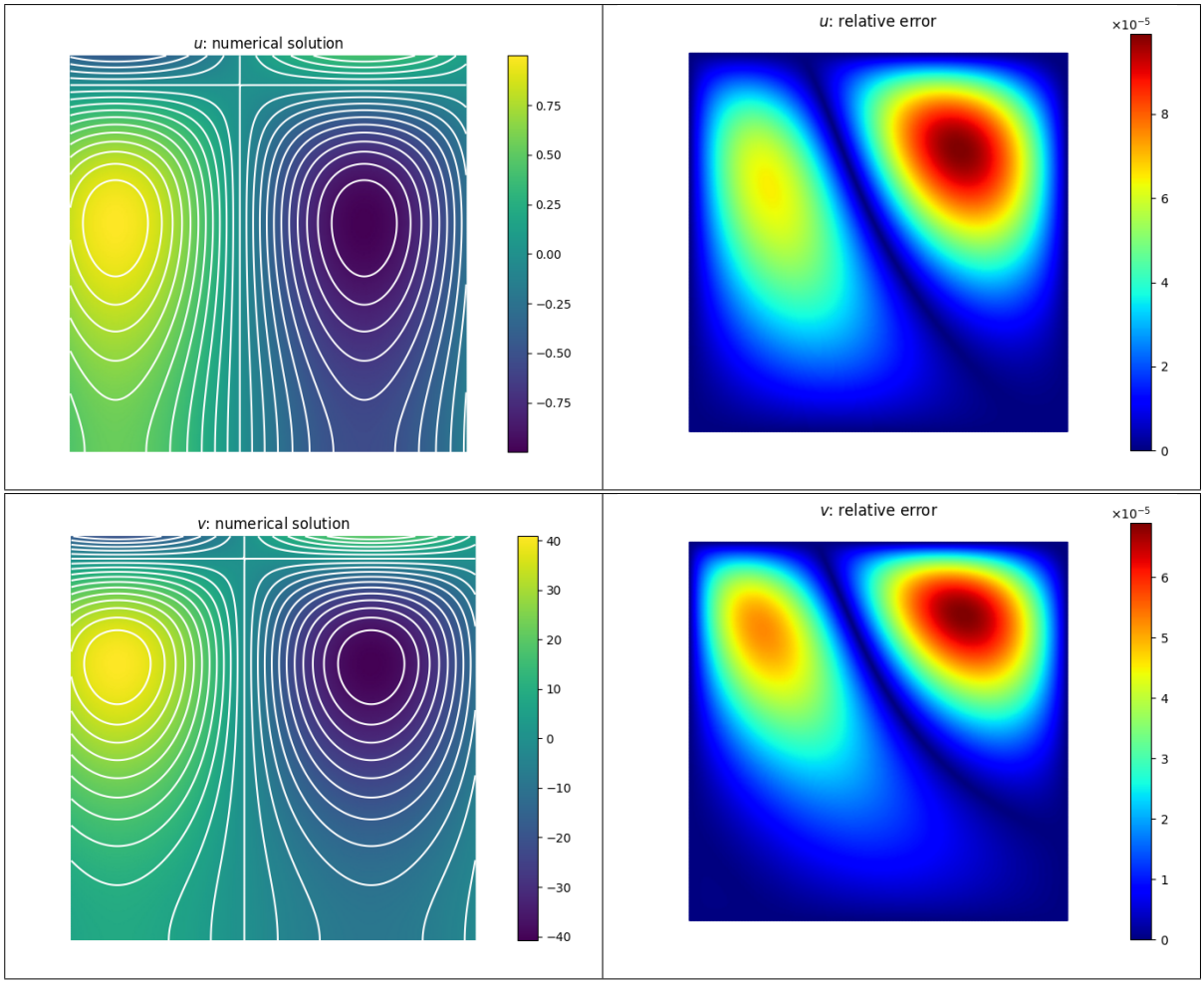


Figure 2.5: 2D biharmonic BVP with SSP boundary conditions, u numerical solution (upper left) with error (upper right) and $v = -\Delta u$ numerical solution (bottom left) with error (bottom right)

One can also compute the orders of convergence by using the `order` function. We use this function in Listing 2.20 and Listing 2.21 on respectively the meshes generated with the `HyperCube` function and the meshes given by `gmsh` with `square4.geo` file. Superconvergence phenomenon with the `HyperCube`

mesh is illustred in Figure 2.6.

Listing 2.20: Computing orders for the biharmonic proble with simply supported plate boundary conditions and with `sol='sin(5*x+1)*cos(3*y**2-1)'`

```
from fc_vfemp1_biharmonic.examples.biharmonic_SSP_square_ex import order
order(sol='sin(5*x+1)*cos(3*y**2-1)')
```

Output

```
------------------------------------------------------------
Solving biharmonic Boundary Value Problems (BVP's) with
        fc_vfemp1_biharmonic package
Copyright (C) 2017 Cuvelier F.
  (LAGA/CNRS/Paris 13 University)
------------------------------------------------------------
     2D biharmonic BVP on [0,1]x[0,1]
     Mesh generated with HyperCube function.

             Laplacian(laplacian(u)) = f

 with boundary conditions
   * Simply Supported Plate (SSP)
      u=gc and Laplacian(u)=gd on [1,2,3,4]
 Exact solution : sin(5*x+1)*cos(3*y**2-1)
------------------------------------------------------------
[ 1/ 4] N=25
   Mesh sizes : nq= 676, nme=  1250, h=5.657e-02
   ndof=1352 - Error : uH1=9.95e-03, uL2=3.41e-03, vH1=9.91e-03, vL2=7.07e-03
[ 2/ 4] N=50
   Mesh sizes : nq= 2601, nme= 5000, h=2.828e-02
   ndof=5202 - Error : uH1=2.51e-03, uL2=8.65e-04, vH1=2.38e-03, vL2=1.79e-03
[ 3/ 4] N=100
   Mesh sizes : nq= 10201, nme= 20000, h=1.414e-02
   ndof=20402 - Error : uH1=6.24e-04, uL2=2.17e-04, vH1=5.90e-04, vL2=4.48e-04
[ 4/ 4] N=150
   Mesh sizes : nq= 22801, nme= 45000, h=9.428e-03
   ndof=45602 - Error : uH1=2.76e-04, uL2=9.65e-05, vH1=2.61e-04, vL2=1.99e-04
```

Listing 2.21: Computing orders for the biharmonic clamped plate problem with `sol='sin(5*x+1)*cos(3*y**2-1)'`

```
from fc_vfemp1_biharmonic.examples.biharmonic_SSP_square_ex import order
order(regular=False, sol='sin(5*x+1)*cos(3*y**2-1)')
```

Output

```
------------------------------------------------------------
Solving biharmonic Boundary Value Problems (BVP's) with
        fc_vfemp1_biharmonic package
Copyright (C) 2017 Cuvelier F.
  (LAGA/CNRS/Paris 13 University)
------------------------------------------------------------
     2D biharmonic BVP on [0,1]x[0,1]
     Mesh generated with gmsh and square4.geo file.

             Laplacian(laplacian(u)) = f

 with boundary conditions
   * Simply Supported Plate (SSP)
      u=gc and Laplacian(u)=gd on [1,2,3,4]
 Exact solution : sin(5*x+1)*cos(3*y**2-1)
------------------------------------------------------------
[ 1/ 4] N=25
   Mesh sizes : nq= 891, nme=  1680, h=5.510e-02
   ndof=1782 - Error : uH1=9.77e-03, uL2=2.01e-03, vH1=1.25e-02, vL2=3.59e-03
[ 2/ 4] N=50
   Mesh sizes : nq= 3436, nme= 6670, h=2.947e-02
   ndof=6872 - Error : uH1=4.21e-03, uL2=5.28e-04, vH1=5.90e-03, vL2=9.08e-04
[ 3/ 4] N=100
   Mesh sizes : nq= 13469, nme= 26536, h=1.503e-02
   ndof=26938 - Error : uH1=1.99e-03, uL2=1.30e-04, vH1=2.80e-03, vL2=2.27e-04
[ 4/ 4] N=150
   Mesh sizes : nq= 30081, nme= 59560, h=9.872e-03
   ndof=60162 - Error : uH1=1.30e-03, uL2=5.86e-05, vH1=1.89e-03, vL2=1.03e-04
```

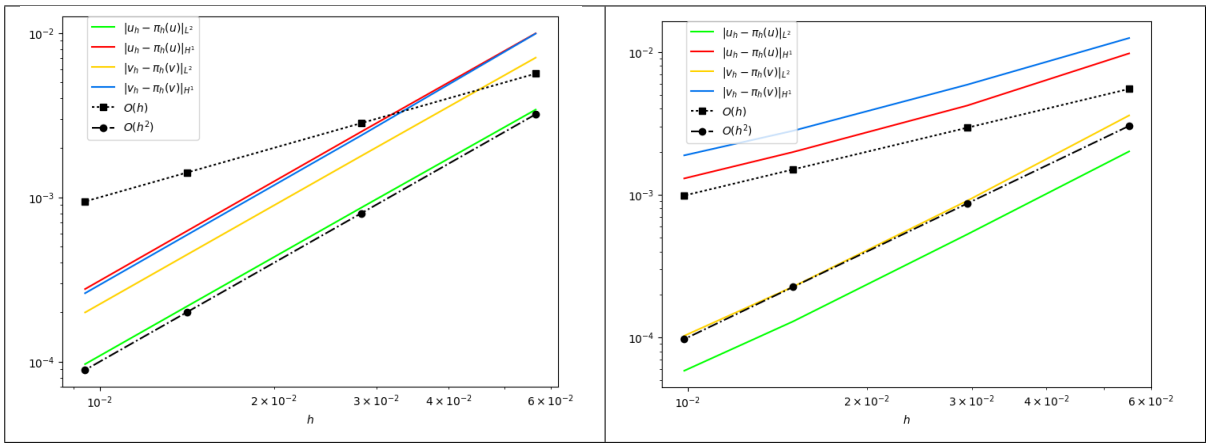


Figure 2.6: 2D biharmonic BVP with (SSP) boundary conditions, order of convergence of u and $v = -\Delta u$ for `HyperCube` mesh (left) and `square4` mesh (right). Superconvergence phenomenon with the `HyperCube` mesh.

2.3.4 Simply supported plate problems on the unit disk

For better understanding it is better to read in advance the section 2.3.1.

The mesh of the unit disk can be obtained by using `gmsh` and a given *.geo* file. This can be done entirely by using FC-OOGMSH and FC-SIMESH packages, and the *disk4bounds.geo* file. The complete code is given in Listing ?? with a graphical representation of the obtained mesh.

In file `biharmonic_SSP_disk_ex.py`[4]) the `run` function can solve a (generic) simply supported plate problem, *Usual* BVP 3, on the unit disk for a given *exact* solution u_{ex} as a string (default is `'sin(x)*sin(y)'`).

Now we will set the simply supported plate boundary conditions on this problem. On each boundary we can compute $g_a = u_{\text{ex}}$ and $g_b = -v$.

Part of the source code which sets the biharmonic problem with the 4 boundary conditions is given in Listing 2.22

```
bvp=blib.BiharmonicBVP(Th, f=f)
for lab in [1,2,3,4]:
  blib.setSimplySupportedPlate(bvp,lab,[u,v])
```

Listing 2.22: Biharmonic BVP with Simply Supported Plate (SSP) boundary conditions on the unit disk

In Listing 2.23, we solve the simply supported plate problem with exact solution $u_{\text{ex}} = \sin(5x + 1)\cos(3y^2 - 1)$ and the output is presented. The graphical results are given in Figure 2.7

[4] directory `fc_vfemp1_biharmonic/examples/`

Listing 2.23: Running biharmonic simply supported plate problem with `sol='sin(5*x+1)*cos(3*y**2-1)'` on the unit disk

```
from fc_vfemp1_biharmonic.examples.biharmonic_SSP_disk_ex import run
res=run(N=150,sol='sin(5*x+1)*cos(3*y**2-1)')
```

Output

```
-------------------------------------------------------------
Solving biharmonic Boundary Value Problems (BVP's) with
        fc_vfemp1_biharmonic package
Copyright (C) 2017 Cuvelier F.
  (LAGA/CNRS/Paris 13 University)
-------------------------------------------------------------
     2D biharmonic BVP on unit disk
     Mesh generated with gmsh and disk4bounds.geo file.

             Laplacian(laplacian(u)) = f

 with boundary conditions
   * Simply Supported Plate (SSP)
      u=gc and Laplacian(u)=gd on [1,2,3,4]
 Exact solution : sin(5*x+1)*cos(3*y**2-1)
-------------------------------------------------------------
*** Setting the mesh using gmsh and disk4bounds.geo file.
    Mesh sizes : nq=94571, nme=188196, h=1.060e-02
*** Setting the BVP
*** Solving the BVP
*** Computing relative errors
  u: errL2=2.3800e-04, errH1=1.3700e-03, errInf=2.7991e-04
  v: errL2=1.1341e-04, errH1=1.7660e-03, errInf=2.7009e-04
*** Plotting solutions
```

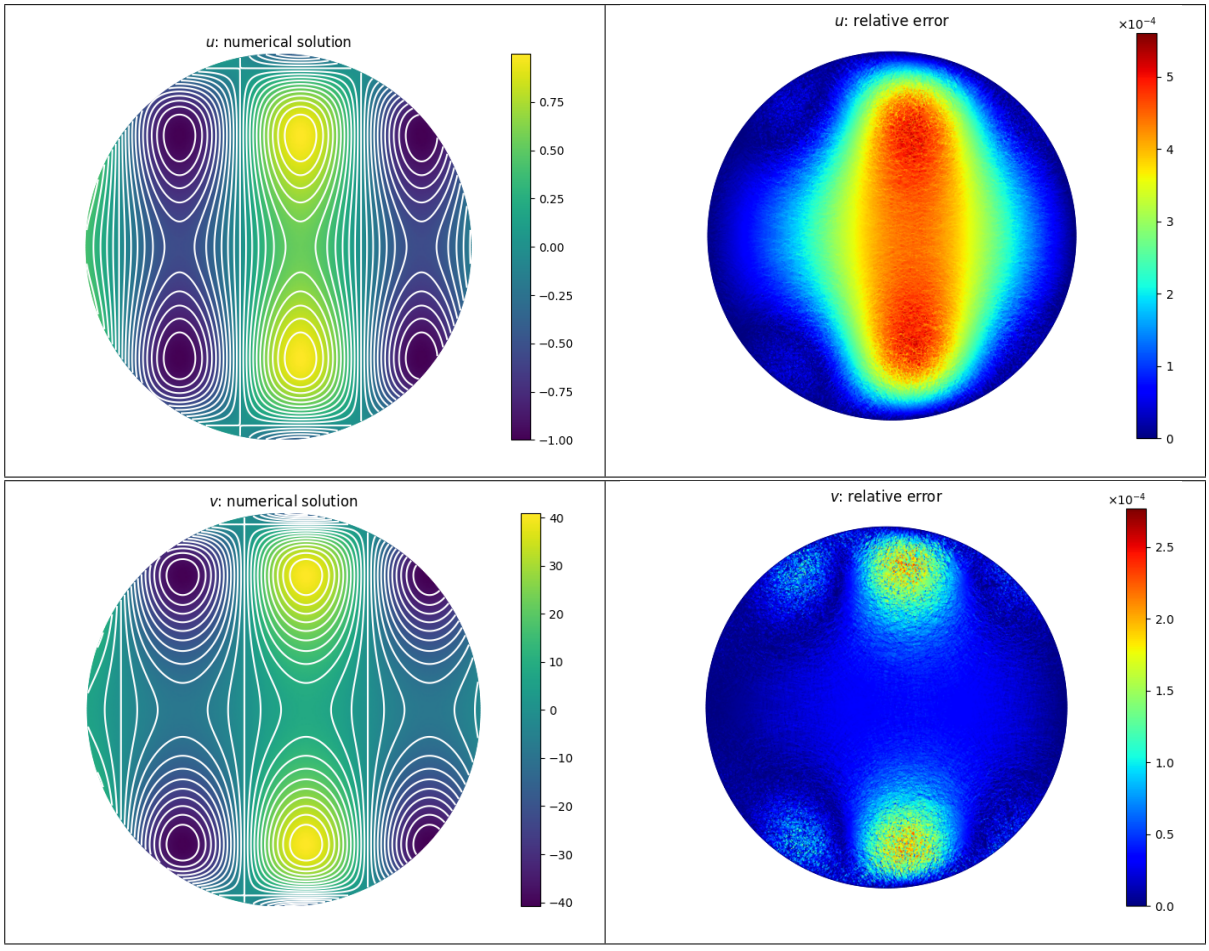


Figure 2.7: Biharmonic BVP with SSP boundary conditions on the unit disk, u numerical solution (upper left) with error (upper right) and $v = -\Delta u$ numerical solution (bottom left) with error (bottom right)

One can also compute the orders of convergence by using the `order` function. We use this function in Listing 2.24 and the order are represented in Figure 2.8.

Listing 2.24: Computing orders for the biharmonic clamped plate problem with `sol='sin(5*x+1)*cos(3*y**2-1)'` on the unit disk

```
from fc_vfemp1_biharmonic.examples.biharmonic_SSP_disk_ex import order
order(sol='sin(5*x+1)*cos(3*y**2-1)')
```

Output

```
------------------------------------------------------------
Solving biharmonic Boundary Value Problems (BVP's) with
        fc_vfemp1_biharmonic package
Copyright (C) 2017 Cuvelier F.
  (LAGA/CNRS/Paris 13 University)
------------------------------------------------------------
     2D biharmonic BVP on unit disk
     Mesh generated with gmsh and disk4bounds.geo file.

             Laplacian(laplacian(u)) = f

 with boundary conditions
   * Simply Supported Plate (SSP)
       u=gc and Laplacian(u)=gd on [1,2,3,4]
 Exact solution : sin(5*x+1)*cos(3*y**2-1)
------------------------------------------------------------
[ 1/ 6] N=25
   Mesh sizes : nq= 2766, nme= 5370, h=5.760e-02
   ndof=5532 - Error : uH1=1.16e-02, uL2=8.13e-03, vH1=1.18e-02, vL2=3.90e-03
[ 2/ 6] N=75
   Mesh sizes : nq= 23549, nme= 46624, h=2.127e-02
   ndof=47098 - Error : uH1=2.93e-03, uL2=9.83e-04, vH1=3.61e-03, vL2=4.63e-04
[ 3/ 6] N=100
   Mesh sizes : nq= 42319, nme= 84004, h=1.539e-02
   ndof=84638 - Error : uH1=2.08e-03, uL2=5.21e-04, vH1=2.62e-03, vL2=2.51e-04
[ 4/ 6] N=150
   Mesh sizes : nq= 94571, nme= 188196, h=1.060e-02
   ndof=189142 - Error : uH1=1.37e-03, uL2=2.38e-04, vH1=1.77e-03, vL2=1.13e-04
[ 5/ 6] N=200
   Mesh sizes : nq=167069, nme= 332876, h=7.726e-03
   ndof=334138 - Error : uH1=1.02e-03, uL2=1.35e-04, vH1=1.32e-03, vL2=6.40e-05
[ 6/ 6] N=250
   Mesh sizes : nq=261161, nme= 520748, h=6.306e-03
   ndof=522322 - Error : uH1=8.14e-04, uL2=8.48e-05, vH1=1.05e-03, vL2=4.07e-05
```

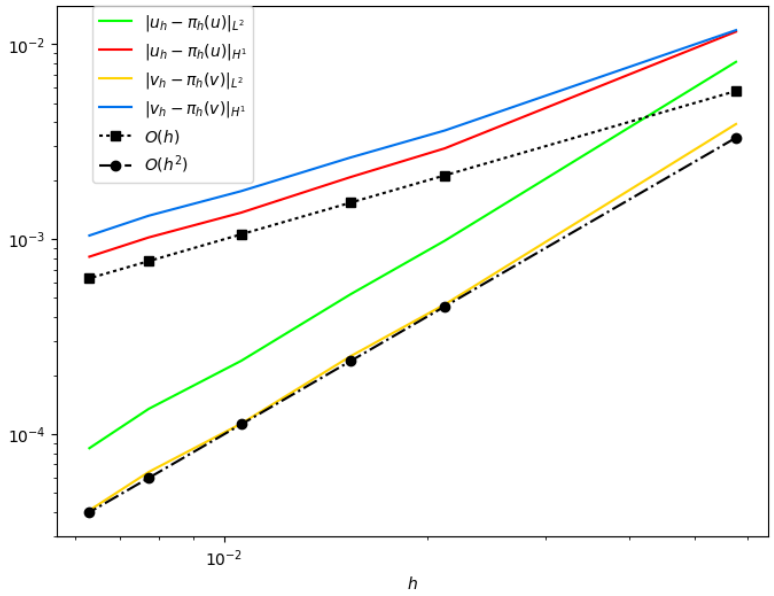


Figure 2.8: 2D biharmonic BVP with (SSP) boundary conditions on the unit disk, order of convergence of u and $v = -\Delta u$.

2.4 Examples

2.4.1 Mixed boundary conditions on a disk with 5 holes

The domain Ω and its boundaries generated by gmsh are represented in Figure 2.9.

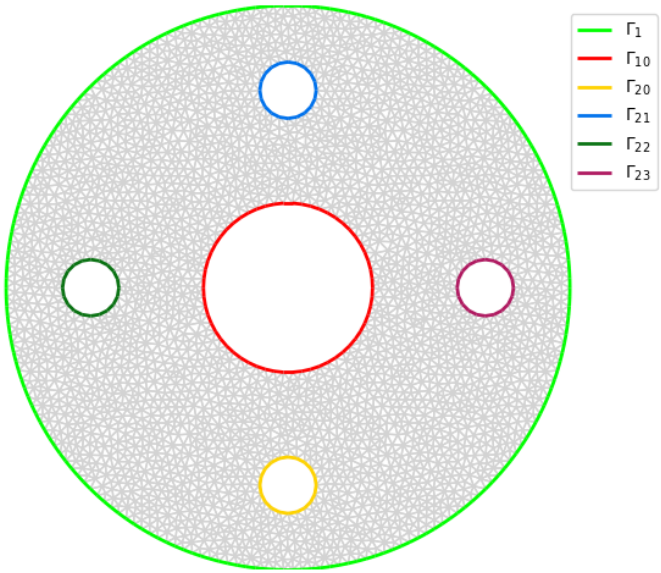


Figure 2.9: Disk with 5 holes : gmsh mesh

We study the following biharmonic BVP with mixed boundary conditions :

***Usual* BVP 5 : Mixed boundary conditions on a disk with 5 holes**

Find u such that

$$\begin{cases} \Delta^2 u & = f & \text{in } \Omega \\ \text{(CP)} \quad u & = 0 \quad \textbf{and} \quad \frac{\partial u}{\partial \boldsymbol{n}} = 0, & \text{on } \Gamma_1 \\ \text{(CH)} \quad \frac{\partial \Delta u}{\partial \boldsymbol{n}} & = 0 \quad \textbf{and} \quad \frac{\partial u}{\partial \boldsymbol{n}} = 0, & \text{on } \Gamma_{10} \\ \text{(SSP)} \quad u & = 1 \quad \textbf{and} \quad \Delta u = 0, & \text{on } \Gamma_{20} \cup \Gamma_{21} \cup \Gamma_{22} \cup \Gamma_{23} \end{cases} \tag{2.24}$$

Part of the source code (file `fc_vfemp1_biharmonic/examples/biharmonic_Mixed_disk5holes_01.py`) is given in Listing 2.25

```
meshfile=gmsh.buildmesh2d(geoFile,N, force=True,verbose=0)
Th=siMesh(meshfile)
bvp=blib.BiharmonicBVP(Th, f=0)
blib.setClampedPlate(bvp,1,[0,0])
blib.setCahnHilliard(bvp,10,[0,0])
for lab in [20,21,22,23]:
   blib.setSimplySupportedPlate(bvp,lab,[1,0])
U=bvp.solve(split=True)
```

Listing 2.25: Mixed boundary conditions on a disk with 5 holes

We represent in Figure 2.10 the numerical error for u computation on the `gmsh` mesh with $n_q = 83324$ and $n_{me} = 165044$.

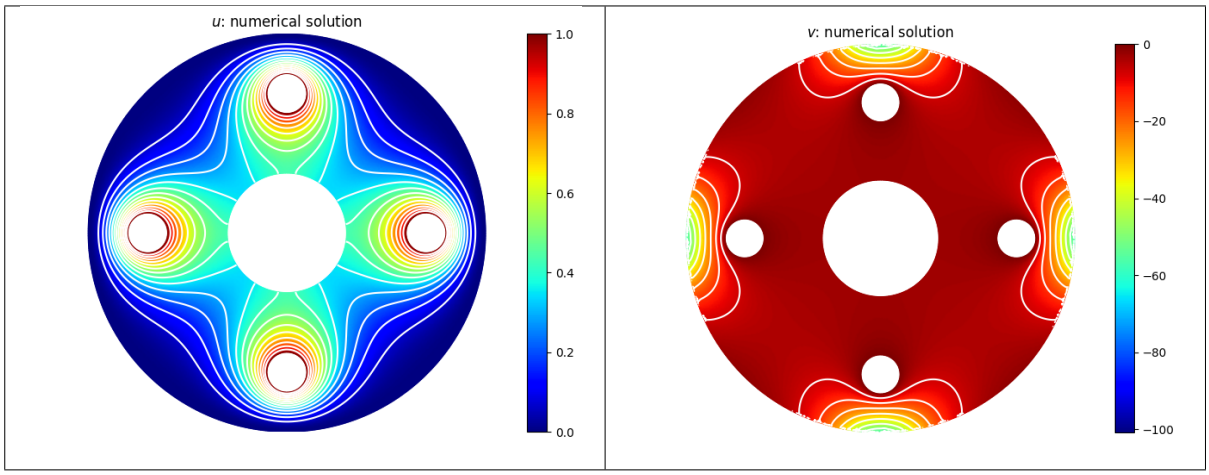


Figure 2.10: mixed boundary conditions on a disk with 5 holes, `(CP)` $(0,0)$ on Γ_1, `(CH)` $(0,0)$ on Γ_{10}, `(SSP)` $(1,0)$ on $\Gamma_{20} \cup \Gamma_{21} \cup \Gamma_{22} \cup \Gamma_{23}$. u numerical solution (left) and $v = -\Delta u$ numerical solution (right)

We represent in Figure 2.11 the same problem except on Γ_{10} : we set a `(CP)` boundary condition instead of a `(CH)` boundary condition:

$$\text{(CP)} \quad u = 0 \quad \textbf{and} \quad \frac{\partial u}{\partial \boldsymbol{n}} = 0, \quad \text{on } \Gamma_{10}.$$

Part of the source code (file `biharmonic_Mixed_disk5holes_01.py`[5]) is given in Listing 2.26

```
meshfile=gmsh.buildmesh2d(geoFile,N, force=True,verbose=0)
Th=siMesh(meshfile)
bvp=blib.BiharmonicBVP(Th, f=0)
blib.setClampedPlate(bvp,1,[0,0])
blib.setClampedPlate(bvp,10,[0,0])
for lab in [20,21,22,23]:
   blib.setSimplySupportedPlate(bvp,lab,[1,0])
U=bvp.solve(split=True)
```

Listing 2.26: Mixed boundary conditions on a disk with 5 holes

The source code is given in file `biharmonic_Mixed_disk5holes_01.py`[6]

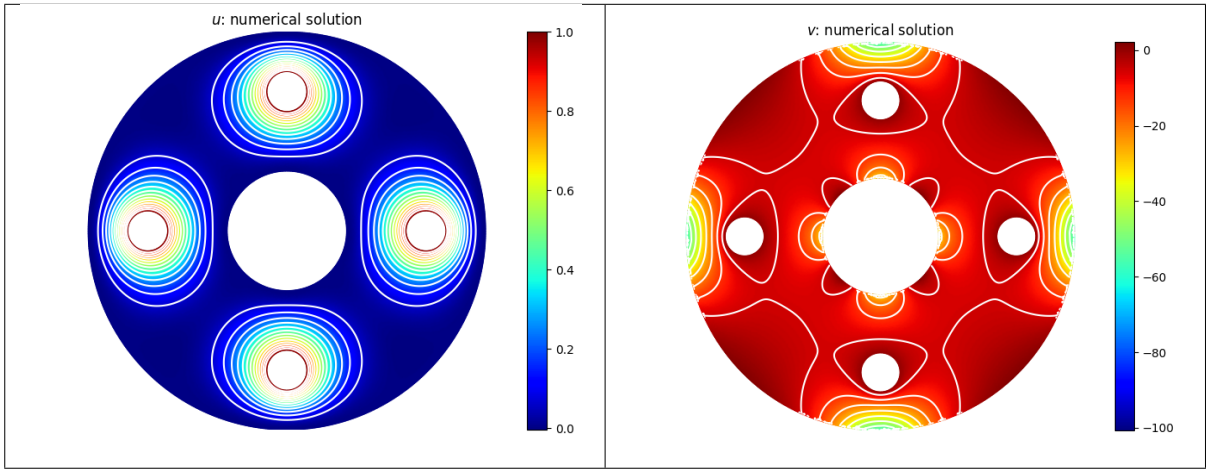


Figure 2.11: mixed boundary conditions on a disk with 5 holes, `(CP)` $(0,0)$ on $\Gamma_1 \cup \Gamma_{10}$, `(SSP)` $(1,0)$ on $\Gamma_{20} \cup \Gamma_{21} \cup \Gamma_{22} \cup \Gamma_{23}$. u numerical solution (left) and $v = -\Delta u$ numerical solution (right)

[5] directory `fc_vfemp1_biharmonic/examples/`
[6] directory `fc_vfemp1_biharmonic/examples/`

2.4.2 2D Biharmonic BVP with (CP) and (CH) boundary conditions

This example comes from [6](p.3104). Let $\Omega = [-1,1]\times[0,1], \Gamma_A = \{[-1,0]\times\{y=0\}\}\cup\{\{x=1\}\times[0,1]\}$, and $\Gamma_B = \Gamma\backslash\Gamma_A$. The domain Ω and its boundaries generated by gmsh are represented in Figure 2.12. With this mesh, we have $\Gamma_A = \Gamma_1 \cup \Gamma_3$ and $\Gamma_B = \Gamma_2 \cup \Gamma_4 \cup \Gamma_5$.

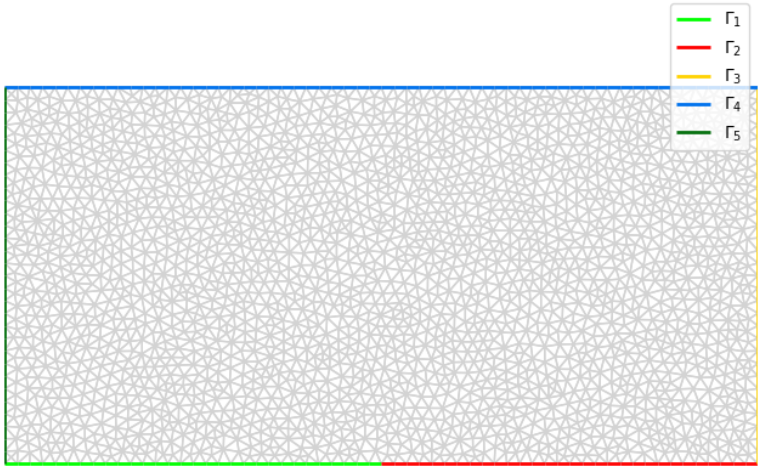


Figure 2.12: 2D biharmonic BVP domain : gmsh mesh

We study the biharmonic BVP with mixed boundary conditions, clamped plate (CP) on Γ_A and Cahn-Hilliard (CH) on Γ_B.

***Usual* BVP 6 : (CP)-(CH) 2D biharmonic problem**

Find u such that

$$\begin{cases} \Delta^2 u & = f \text{ in } \Omega \\ \text{(CP)} \quad u & = g_A \text{ and } \frac{\partial u}{\partial \boldsymbol{n}} = 0, \text{ on } \Gamma_A = \Gamma_1 \cup \Gamma_3 \\ \text{(CH)} \quad \frac{\partial \Delta u}{\partial \boldsymbol{n}} & = 0 \text{ and } \frac{\partial u}{\partial \boldsymbol{n}} = 0, \text{ on } \Gamma_B = \Gamma_2 \cup \Gamma_4 \cup \Gamma_5 \end{cases} \tag{2.25}$$

where $g_A = 0$ on Γ_1 and $g_A = 1$ on Γ_3.

Part of the source code (file `fc_vfemp1_biharmonic/examples/biharmonic_CP_CH_rectangle01_crack01.py`) is given in Listing 2.27

```
meshfile=gmsh.buildmesh2d(geoFile,N, force=True,verbose=0)
Th=siMesh(meshfile)
bvp=blib.BiharmonicBVP(Th, f=0)
blib.setClampedPlate(bvp,1,[0,0])
blib.setClampedPlate(bvp,3,[1,0])
U=bvp.solve(split=True)
```

Listing 2.27: 2D biharmonic BVP with (CP)-(CH) boundary conditions with crack singularities

We represent in Figure 2.13 the numerical error for u computation on the gmsh mesh with $n_q = 59825$ and $n_{me} = 118748$.

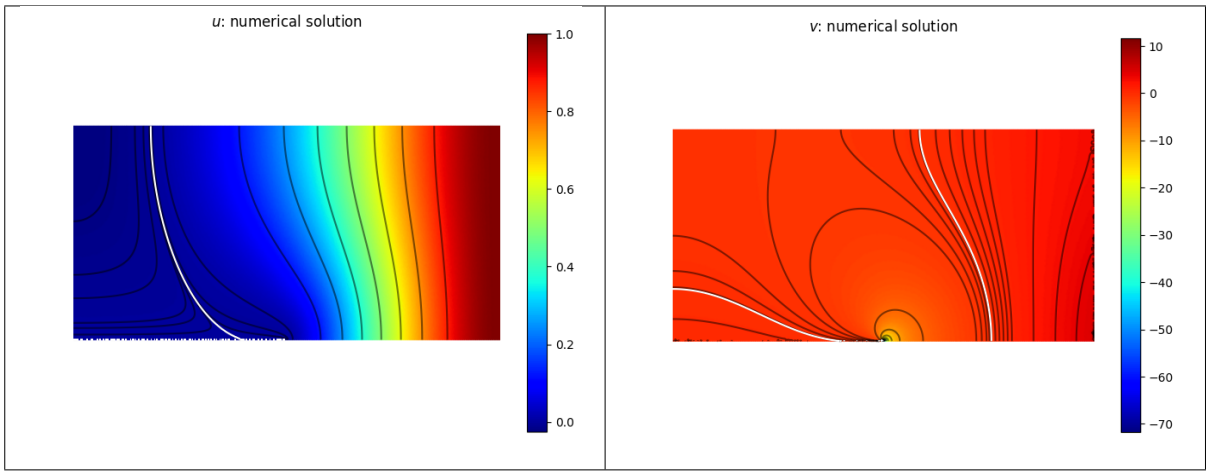


Figure 2.13: 2D biharmonic BVP with (CP)-(CH) boundary conditions, u numerical solution (left) and $v = -\Delta u$ numerical solution (right)

Chapter 3

Pseudo-biharmonic Boundary Value Problems

3.1 Clamped grid problem

In [7] page 74, *the operator* $\frac{\partial^4 u}{\partial x^4} + \frac{\partial^4 u}{\partial y^4}$ *is used as a model of an elastic medium consisting in two sets of intertwined (not glued) perpendicular fibers running in Cartesian directions (Figure 3.1). The main assumption here is that those sets of fibers are connected in such a way that the vertical positions coincide but there is no connection that forces a torsion in the fibers.*

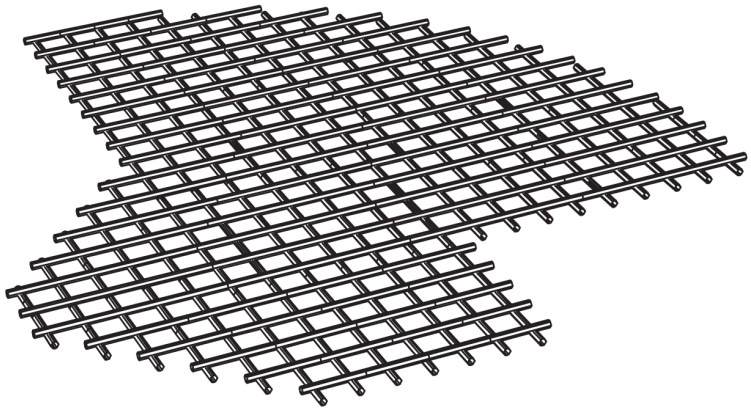


Figure 3.1: A fragment of an elastic grid.

The Clamped grid problem is

Usual BVP 7 : 2D clamped grid problem

Find u such that

$$\begin{cases} \frac{\partial^4 u}{\partial x^4} + \frac{\partial^4 u}{\partial y^4} &= f \quad \text{in } \Omega \\ u = \frac{\partial u}{\partial \boldsymbol{n}} &= 0 \quad \text{on } \Gamma \end{cases} \tag{3.1}$$

We have

$$\frac{\partial^4}{\partial x^4} + \frac{\partial^4}{\partial y^4} = \left(-\Delta - \sqrt{2}\frac{\partial^2}{\partial x \partial y}\right)\left(-\Delta + \sqrt{2}\frac{\partial^2}{\partial x \partial y}\right)$$

and problem (3.1) may be split into :

Usual BVP 8 : 2D clamped grid problem splitting

Find u and v such that

$$\begin{cases} -\Delta v - \sqrt{2}\frac{\partial^2 v}{\partial x \partial y} &= f, \quad \text{in } \Omega \\ -\Delta u + \sqrt{2}\frac{\partial^2 u}{\partial x \partial y} &= v, \quad \text{in } \Omega \\ u = \frac{\partial u}{\partial \boldsymbol{n}} &= 0, \quad \text{on } \Gamma \end{cases} \tag{3.2}$$

The two first equations can be equivalently written as

$$\mathcal{G}\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} f \\ 0 \end{pmatrix} \quad \text{or} \quad \mathcal{K}\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} 0 \\ f \end{pmatrix} \tag{3.3}$$

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

$$\mathcal{G} = \begin{pmatrix} 0 & \mathcal{L}_{\mathbb{E},\mathbf{0},\mathbf{0},0} \\ \mathcal{L}_{\mathbb{F},\mathbf{0},\mathbf{0},0} & \mathcal{L}_{\mathbb{O},\mathbf{0},\mathbf{0},-1} \end{pmatrix} \quad \text{and} \quad \mathcal{K} = \begin{pmatrix} \mathcal{L}_{\mathbb{F},\mathbf{0},\mathbf{0},0} & \mathcal{L}_{\mathbb{O},\mathbf{0},\mathbf{0},-1} \\ \mathcal{L}_{\mathbb{O},\mathbf{0},\mathbf{0},0} & \mathcal{L}_{\mathbb{E},\mathbf{0},\mathbf{0},0} \end{pmatrix} \tag{3.4}$$

with

$$\mathbb{E} = \begin{pmatrix} 1 & \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & 1 \end{pmatrix} \quad \text{and} \quad \mathbb{F} = \begin{pmatrix} 1 & -\frac{\sqrt{2}}{2} \\ -\frac{\sqrt{2}}{2} & 1 \end{pmatrix}.$$

We choose these expressions for $\mathbb{E}$ and $\mathbb{F}$ to preserve symetry.

Let $\boldsymbol{w} = (u, v)$. From (3.3), the components of the conormal derivative of $\boldsymbol{w}$ defined in (1.17) are given by

$$\begin{aligned} \frac{\partial \boldsymbol{w}}{\partial n_{\mathcal{K}_1}} &\overset{\text{def}}{=} \sum_{\beta=1}^{2} \frac{\partial \boldsymbol{w}_\beta}{\partial n_{\mathcal{K}_{1,\beta}}} = \sum_{\beta=1}^{2} \left\langle \mathbb{A}^{1,\beta} \boldsymbol{\nabla} \boldsymbol{w}_\beta, \boldsymbol{n} \right\rangle - \left\langle \boldsymbol{b}^{1,\beta} \boldsymbol{u}_\beta, \boldsymbol{n} \right\rangle \\ &= \left\langle \mathbb{F} \boldsymbol{\nabla} \boldsymbol{w}_1, \boldsymbol{n} \right\rangle = \left\langle \mathbb{F} \boldsymbol{\nabla} u, \boldsymbol{n} \right\rangle = \frac{\partial \boldsymbol{w}}{\partial n_{\mathcal{G}_2}} \end{aligned} \tag{3.5}$$

and

$$\begin{aligned} \frac{\partial \boldsymbol{w}}{\partial n_{\mathcal{K}_2}} &\overset{\text{def}}{=} \sum_{\beta=1}^{2} \frac{\partial \boldsymbol{w}_\beta}{\partial n_{\mathcal{K}_{2,\beta}}} = \sum_{\beta=1}^{2} \left\langle \mathbb{A}^{2,\beta} \boldsymbol{\nabla} \boldsymbol{w}_\beta, \boldsymbol{n} \right\rangle - \left\langle \boldsymbol{b}^{2,\beta} \boldsymbol{u}_\beta, \boldsymbol{n} \right\rangle \\ &= \left\langle \mathbb{E} \boldsymbol{\nabla} \boldsymbol{w}_2, \boldsymbol{n} \right\rangle = \left\langle \mathbb{E} \boldsymbol{\nabla} v, \boldsymbol{n} \right\rangle = \frac{\partial \boldsymbol{w}}{\partial n_{\mathcal{G}_1}} \end{aligned} \tag{3.6}$$

With $\mathcal{G}$ operator one can impose the following boundary conditions

$$\begin{aligned} \boldsymbol{w}_\alpha &= g_\alpha^D && \text{on } \Gamma_\alpha^D, \ \forall \alpha \in [\![1,1]\!], \\ \frac{\partial \boldsymbol{w}}{\partial n_{\mathcal{G}_\alpha}} + a_\alpha^R \boldsymbol{w}_\alpha &= g_\alpha^R && \text{on } \Gamma_\alpha^R, \ \forall \alpha \in [\![1,2]\!] \end{aligned}$$

and we obtain

$$\begin{aligned} u &= g_1^D && \text{on } \Gamma_1^D, \text{ and } && v = g_2^D && \text{on } \Gamma_2^D, \\ \langle \mathbb{E} \boldsymbol{\nabla} v, \boldsymbol{n} \rangle + a_1^R u &= g_1^R && \text{on } \Gamma_1^R, \text{ and } && \langle \mathbb{F} \boldsymbol{\nabla} u, \boldsymbol{n} \rangle + a_2^R v = g_2^R && \text{on } \Gamma_2^R \end{aligned}$$

To impose clamped grid boundary condition $u = \frac{\partial u}{\partial \boldsymbol{n}} = 0$ on Γ, we choose $\Gamma_1^D = \Gamma$ (thus $\Gamma_1^R = \varnothing$). As explain in Remark 6.1 of [7], when $u = 0$ on Γ, the tangential derivatives of u, $\frac{\partial u}{\partial \tau} \overset{\text{def}}{=} \boldsymbol{\nabla} u - \langle \boldsymbol{\nabla} u, \boldsymbol{n} \rangle \boldsymbol{n}$, is zero on Γ, and then

$$\boldsymbol{\nabla} u = \langle \boldsymbol{\nabla} u, \boldsymbol{n} \rangle \boldsymbol{n}.$$

So we obtain

$$\langle \mathbb{F} \boldsymbol{\nabla} u, \boldsymbol{n} \rangle = \langle \boldsymbol{\nabla} u, \boldsymbol{n} \rangle \langle \mathbb{F} \boldsymbol{n}, \boldsymbol{n} \rangle = \langle \boldsymbol{\nabla} u, \boldsymbol{n} \rangle (1 - \sqrt{2} n_1 n_2).$$

Since the normal $\boldsymbol{n} = (n_1, n_2) = (\cos\theta, \sin\theta)$ one finds that $n_1 n_2 = \frac{1}{2}\sin(2\theta) \in [-\frac{1}{2}, \frac{1}{2}]$ and then $1 - \sqrt{2} n_1 n_2 \geqslant 1 - \frac{\sqrt{2}}{2} > 0$.

Thus, under the asumption $u = 0$ on Γ, we have on Γ

$$\frac{\partial u}{\partial \boldsymbol{n}} = 0 \iff \frac{\partial \boldsymbol{w}}{\partial n_{\mathcal{G}_2}} \overset{\text{def}}{=} \langle \mathbb{F} \boldsymbol{\nabla} u, \boldsymbol{n} \rangle = 0$$

and we can write

***Vector* BVP 5 : clamped grid problem (3.2) with $\mathcal{G}$ operator**

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

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

With $\mathcal{K}$ operator one can impose the following boundary conditions

$$\begin{aligned} \boldsymbol{w}_\alpha &= g_\alpha^D && \text{on } \Gamma_\alpha^D, \ \forall \alpha \in [\![1,1]\!], \\ \frac{\partial \boldsymbol{w}}{\partial n_{\mathcal{K}_\alpha}} + a_\alpha^R \boldsymbol{w}_\alpha &= g_\alpha^R && \text{on } \Gamma_\alpha^R, \ \forall \alpha \in [\![1,2]\!] \end{aligned}$$

and we obtain

$$\begin{aligned} u &= g_1^D && \text{on } \Gamma_1^D, \ \text{and} \quad v = g_2^D && \text{on } \Gamma_2^D, \\ \langle \mathbb{F} \boldsymbol{\nabla} u, \boldsymbol{n} \rangle + a_1^R u &= g_1^R && \text{on } \Gamma_1^R, \ \text{and} \quad \langle \mathbb{E} \boldsymbol{\nabla} v, \boldsymbol{n} \rangle + a_2^R v = g_2^R && \text{on } \Gamma_2^R. \end{aligned}$$

If we want to impose clamped grid boundary condition $u = \frac{\partial u}{\partial \boldsymbol{n}} = 0$ on Γ the choice of $\mathcal{K}$ operator is not possible : with $u = 0$ on $\Gamma_1^D = \Gamma$, we have $\Gamma_1^R = \varnothing$ and we cannot impose $\frac{\partial u}{\partial \boldsymbol{n}} = 0$ on Γ.

3.2 Applications on a rectangular domain

Let Ω_0 be the rectangular domain $[-1, 6] \times [-1, 1]$ and $f_0 := \exp(-100((x + 0.75)^2 + (y - 0.75)^2))$ be the source term as a concentred load. They are represented in Figure 3.6.

We also denote by $\Omega_{\pi/4}$ and $f_{\pi/4}$ respectively the domain Ω and the function f rotated by $\pi/4$ with $(0, 0)$ as center. They are represented in Figure 3.7.

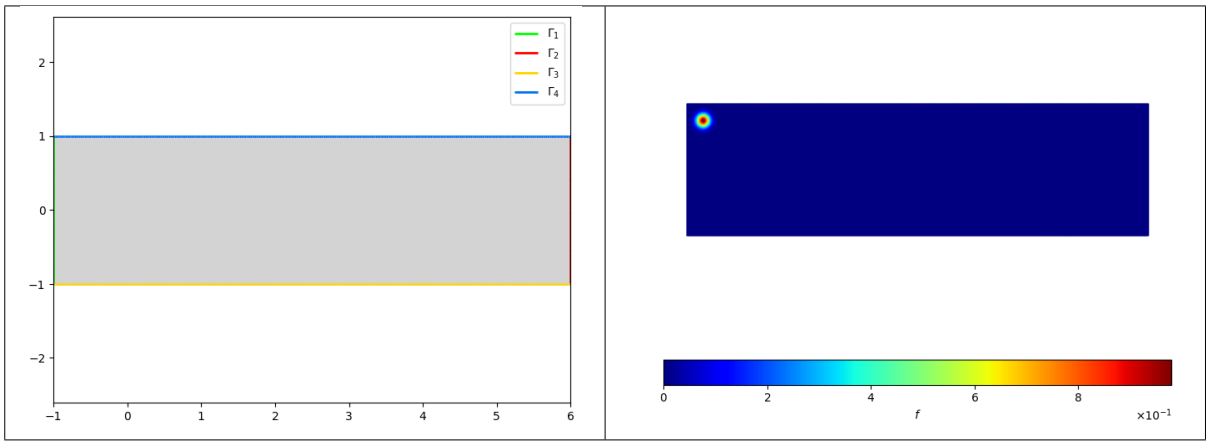


Figure 3.2: Domain $\Omega_0 = [-1, 6] \times [-1, 1]$ (left) and function f_0 (right)

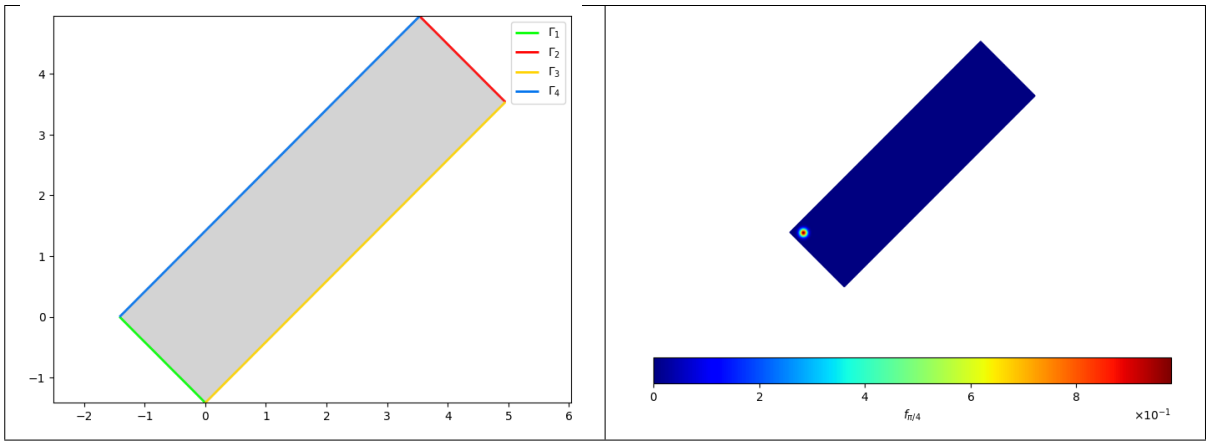


Figure 3.3: Domain $\Omega_{\pi/4}$ (left) and function $f_{\pi/4}$ (right)

We study three problems. The first one will be the classical biharmonic clamped plate problem on Ω_0. The second will be clamped grid problem with the aligned fibers on Ω_0. The last one will be the clamped grid problem with the diagonal fibers on $\Omega_{\pi/4}$.

3.2.1 Classical biharmonic Clamped Plate problem

***Usual* BVP 9 : Clamped plate problem on Ω_0**

Find u such that

$$\begin{cases} \Delta^2 u & = f_0 \quad \text{in } \Omega_0 \\ u = \frac{\partial u}{\partial \boldsymbol{n}} & = 0 \quad \text{on } \Gamma = \partial\Omega_0 \end{cases} \tag{3.7}$$

As seen in section ?? this problem can be written as a vector boundary value problem and is given in *Vector* BVP ??. Part of the Python code used to solve this problem is given in Listing 3.1

```
    print_separator()
    print('     Mesh sizes : nq=%d, nme=%d,
        h=%.3e'%(Th.nq,Th.get_nme(),Th.get_h()))
Hop.H[1][0]=Loperator(dim=2,A=[[1,None],[None,1]])
Hop.H[1][1]=Loperator(dim=2,a0=-1)
pde=PDE(Op=Hop,f=[f,0])
bvp=BVP(Th,pde=pde)
for lab in Th.sThlab[Th.find(d=1)]:
  bvp.setDirichlet(lab,0.0,comps=[0])

if verbose>0:
```

```
if graphics and isModuleFound('fc_simesh_matplotlib'):
```

Listing 3.1: Clamped *plate* problem on $[-1,6]\times[-1,1]$

The complete code is given in file `BVP_ClampedPlate_hypermesh_2D01.py` [1] and can be run with the following python commands:

```
from fc_vfemp1_biharmonic.examples.BVP_ClampedPlate_hypermesh_2D01 import run
res=run(N=50)
```

3.2.2 Clamped grid problem with the aligned fibers

Usual BVP 10 : Clamped grid problem with the aligned fibers on Ω_0

Find u such that

$$\begin{cases} \frac{\partial^4 u}{\partial x^4}+\frac{\partial^4 u}{\partial y^4} &= f_0 \text{ in } \Omega_0 \\ u=\frac{\partial u}{\partial \boldsymbol{n}} &= 0 \text{ on } \Gamma=\partial\Omega_0 \end{cases} \tag{3.8}$$

As seen in section 3.1 this problem can be written as a vector boundary value problem and is given in *Vector* BVP 5. Part of the Python code used to solve this problem is given in Listing 3.2.

```
f=lambda x,y: exp(-100*((x+0.75)**2+(y-0.75)**2))
Th=HyperCube(2,[7*N,2*N],mapping=lambda q: np.array([-1+7*q[0],2*q[1]-1]))
a=2**(1/2)/2
Hop=Hoperator(dim=2,m=2)
Hop.H[0][1]=Loperator(dim=2,A=[[1,a],[a,1]])
Hop.H[1][0]=Loperator(dim=2,A=[[1,-a],[-a,1]])
Hop.H[1][1]=Loperator(dim=2,a0=-1)
pde=PDE(Op=Hop,f=[f,0])
bvp=BVP(Th,pde=pde)
for lab in Th.sThlab[Th.find(d=1)]:
  bvp.setDirichlet(lab,0.0,comps=[0])
U=bvp.solve(split=True)
```

Listing 3.2: Clamped *grid* problem on $[-1,6]\times[-1,1]$

The complete code is given in file `BVP_ClampedAlignedGrid_hypermesh_2D01.py` [2] and can be run with the following python commands:

```
from fc_vfemp1_biharmonic.examples.BVP_ClampedAlignedGrid_hypermesh_2D01 import
    run
res=run(N=50)
```

3.2.3 Clamped grid problem with the diagonal fibers

Usual BVP 11 : Clamped grid problem with the diagonal fibers on $\Omega_{\pi/4}$

Find u such that

$$\begin{cases} \frac{\partial^4 u}{\partial x^4}+\frac{\partial^4 u}{\partial y^4} &= f_{\pi/4} \text{ in } \Omega_{\pi/4} \\ u=\frac{\partial u}{\partial \boldsymbol{n}} &= 0 \text{ on } \Gamma=\partial\Omega_{\pi/4} \end{cases} \tag{3.9}$$

The Python code used to solve this problem is very similar to the previous one given in Listing 3.1: we just have to rotate the function f and the domain. These operations are given in Listing 3.3.

```
theta=pi/4 # For f and mesh rotation
R=[[cos(theta),-sin(theta)],[sin(theta),cos(theta)]]
P=np.dot(R,[-0.75,0.75])
f=lambda x,y:exp(-100*((x - P[0])**2 + (y - P[1])**2))
Th=HyperCube(2,[7*N,2*N],mapping=lambda q:
    np.dot(R,np.array([-1+7*q[0],2*q[1]-1])))
```

Listing 3.3: 2D clamped *grid* BVP on $[-1,6]\times[-1,1]$

[1] directory `fc_vfemp1_biharmonic/examples/`
[2] directory `fc_vfemp1_biharmonic/examples/`

The complete code is given in file `BVP_ClampedDiagonalGrid_hypermesh_2D01.py` [3] and can be run with the following python commands:

```
from fc_vfemp1_biharmonic.examples.BVP_ClampedDiagonalGrid_hypermesh_2D01 import
    run
res=run(N=50)
```

3.2.4 Graphical results

For graphical convenience the numerical solutions on the grid with the diagonal fibers are rotated by $-\pi/4$.

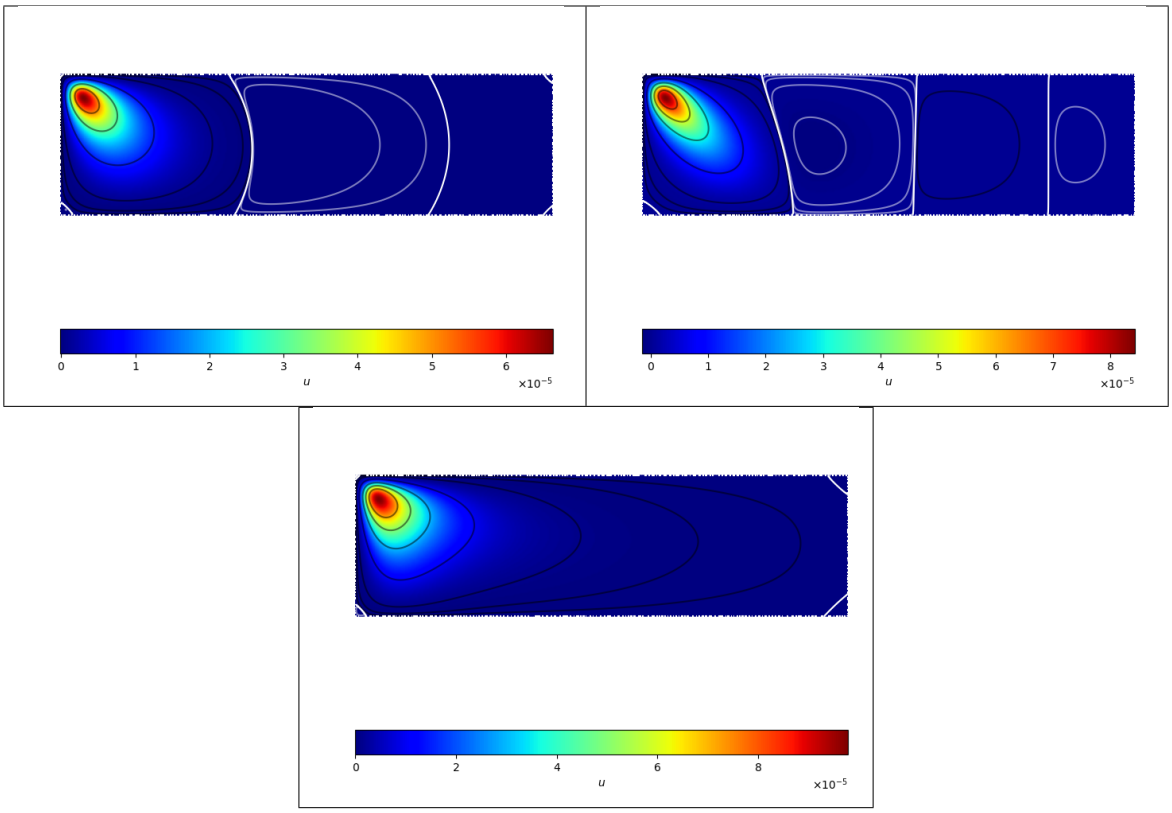


Figure 3.4: Displacement u of, respectively, an isotropic rectangular plate (upper left), grid with the aligned fibers (upper right) and grid with the diagonal fibers (bottom).

[3] directory `fc_vfemp1_biharmonic/examples/`

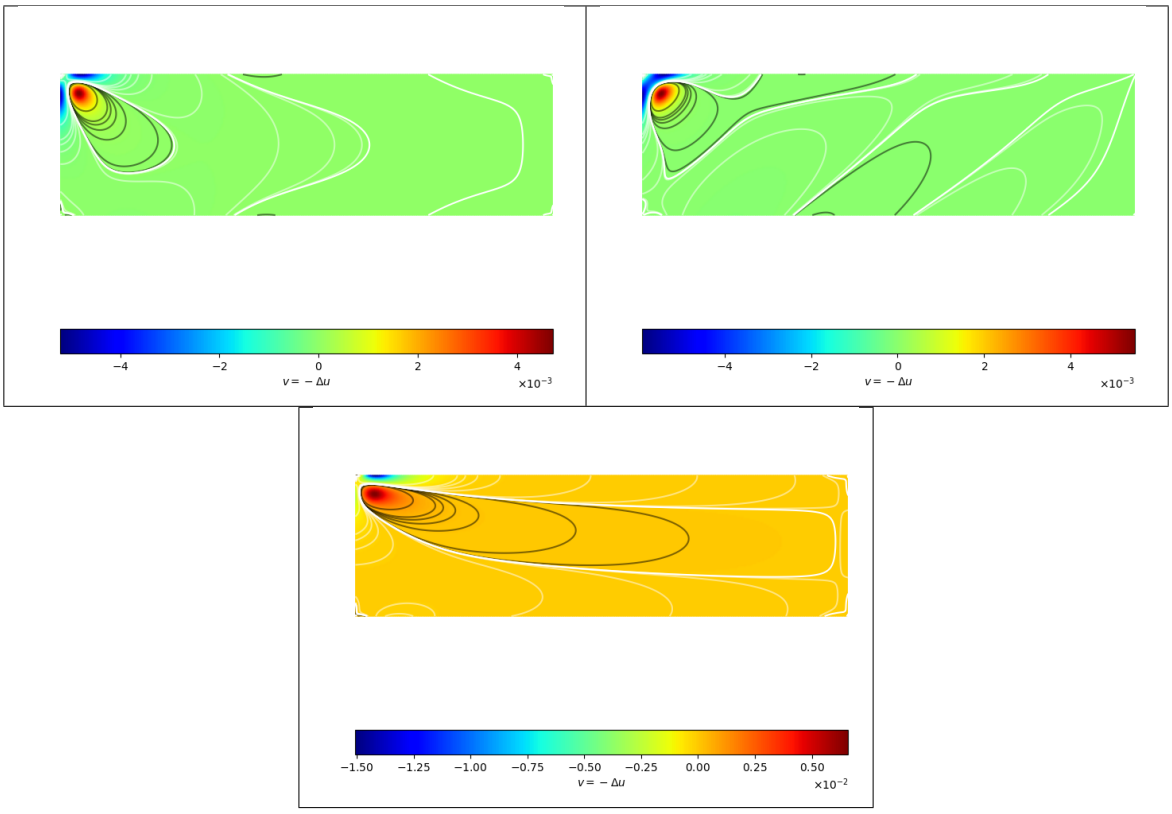


Figure 3.5: $v = -\Delta u$ of, respectively, an isotropic rectangular plate (upper left), grid with the aligned fibers (upper right) and grid with the diagonal fibers (bottom).

3.3 Applications on a pentagonal domain

Let Ω_0 be the pentagon

$$\Omega_0 = \{(x, y) \in [-1, 2] \times [-1, 1] \ : \ y \geqslant -x - 1\}$$

and $f := \exp(-100((x + 0.75)^2 + (y - 0.75)^2))$ be the source term as a concentred load. They are represented in Figure **??**.

We also denote by $\Omega_{\pi/4}$ and $f_{\pi/4}$ respectively the domain Ω and the function f rotated by $\pi/4$ with $(0, 0)$ as center. They are represented in Figure **??**.

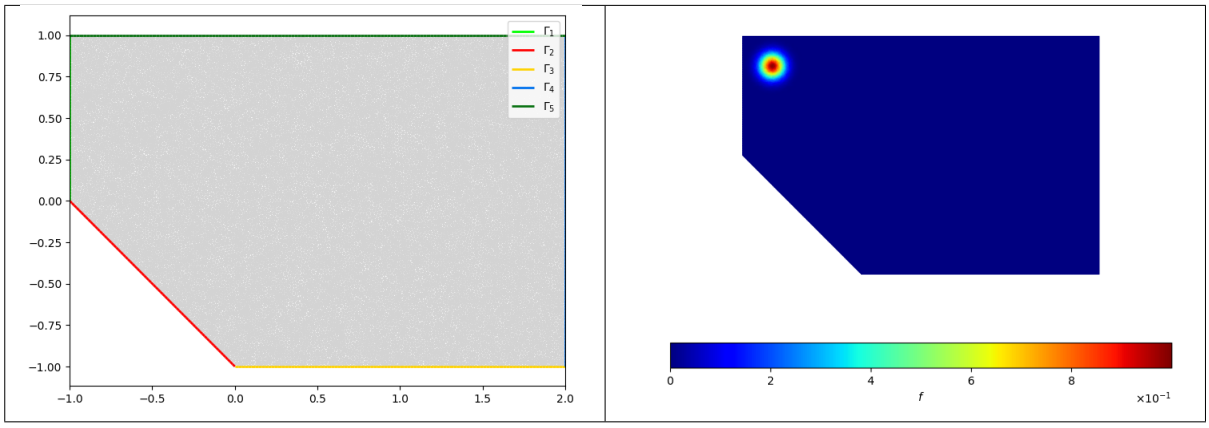


Figure 3.6: Domain Ω_0 (left) and function f_0 (right)

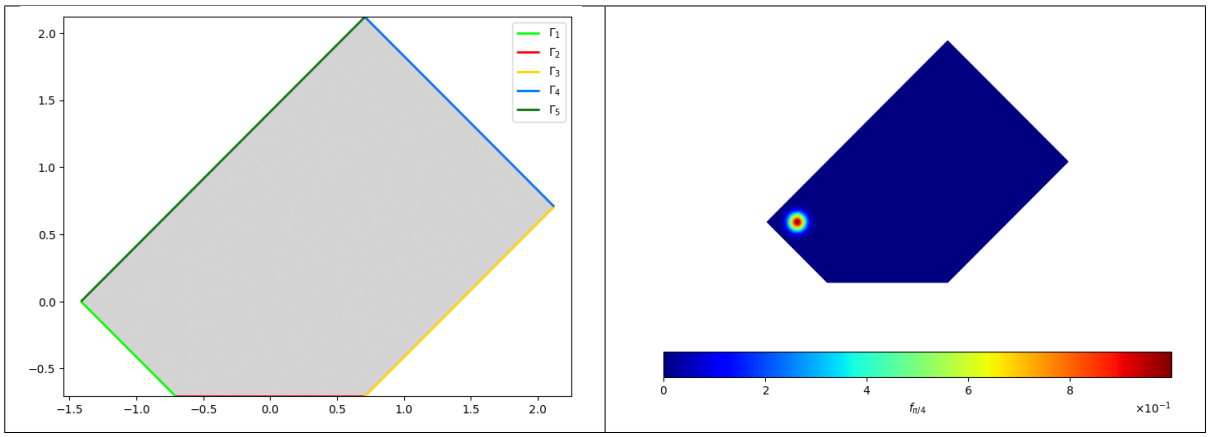


Figure 3.7: Domain $\Omega_{\pi/4}$ (left) and function $f_{\pi/4}$ (right)

We study three problems. The first one will be the classical biharmonic clamped plate problem on Ω_0. The second will be clamped grid problem with the aligned fibers on Ω_0. The last one will be the clamped grid problem with the diagonal fibers on $\Omega_{\pi/4}$.

3.3.1 Classical biharmonic Clamped Plate problem

Usual BVP 12 : Clamped plate problem on Ω_0

Find u such that

$$\begin{cases} \Delta^2 u & = f_0 \ \text{ in } \Omega_0 \\ u = \frac{\partial u}{\partial \boldsymbol{n}} & = 0 \ \text{ on } \Gamma = \partial\Omega_0 \end{cases} \tag{3.10}$$

As seen in section ?? this problem can be written as a vector boundary value problem and is given in *Vector* BVP ??. Part of the Python code used to solve this problem is given in Listing 3.4.

```
f=lambda x,y: exp(-100*((x+0.75)**2+(y-0.75)**2))
meshfile=gmsh.buildmesh2d(geoFile,N, force=True,verbose=0)
Th=siMesh(meshfile)
Hop=Hoperator(dim=2,m=2)
Hop.H[0][1]=Loperator(dim=2,A=[[1,None],[None,1]])
Hop.H[1][0]=Loperator(dim=2,A=[[1,None],[None,1]])
Hop.H[1][1]=Loperator(dim=2,a0=-1)
pde=PDE(Op=Hop,f=[f,0])
bvp=BVP(Th,pde=pde)
for lab in Th.sThlab[Th.find(d=1)]:
  bvp.setDirichlet(lab,0.0,comps=[0])
U=bvp.solve(split=True)
```

Listing 3.4: Clamped *plate* problem on pentagonal domain

The complete code is given in file `BVP_ClampedPlate_pentagonal01_2D01.py` [4] and can be run with the following python commands:

```
from fc_vfemp1_biharmonic.examples.BVP_ClampedPlate_pentagonal01_2D01 import run
res=run(N=50)
```

[4] directory `fc_vfemp1_biharmonic/examples/`

3.3.2 Clamped grid problem with the aligned fibers

Usual BVP 13 : Clamped grid problem with the aligned fibers on Ω_0

Find u such that

$$\begin{cases} \frac{\partial^4 u}{\partial x^4} + \frac{\partial^4 u}{\partial y^4} &= f_0 \text{ in } \Omega_0 \\ u = \frac{\partial u}{\partial \boldsymbol{n}} &= 0 \text{ on } \Gamma = \partial\Omega_0 \end{cases} \tag{3.11}$$

As seen in section 3.1 this problem can be written as a vector boundary value problem and is given in *Vector* BVP 5. Part of the Python code used to solve this problem is given in Listing 3.5.

```
f=lambda x,y: exp(-100*((x+0.75)**2+(y-0.75)**2))
meshfile=gmsh.buildmesh2d(geoFile,N, force=True,verbose=0)
Th=siMesh(meshfile)
a=2**(1/2)/2
Hop=Hoperator(dim=2,m=2)
Hop.H[0][1]=Loperator(dim=2,A=[[1,a],[a,1]])
Hop.H[1][0]=Loperator(dim=2,A=[[1,-a],[-a,1]])
Hop.H[1][1]=Loperator(dim=2,a0=-1)
pde=PDE(Op=Hop,f=[f,0])
bvp=BVP(Th,pde=pde)
for lab in Th.sThlab[Th.find(d=1)]:
  bvp.setDirichlet(lab,0.0,comps=[0])
U=bvp.solve(split=True)
```

Listing 3.5: Clamped *grid* problem on a pentagon

The complete code is given in file `BVP_ClampedAlignedGrid_pentagonal01_2D01.py` [5] and can be run with the following python commands:

```
from fc_vfemp1_biharmonic.examples.BVP_ClampedAlignedGrid_pentagonal01_2D01
    import run
res=run(N=50)
```

3.3.3 Clamped grid problem with the diagonal fibers

Usual BVP 14 : Clamped grid problem with the diagonal fibers on $\Omega_{\pi/4}$

Find u such that

$$\begin{cases} \frac{\partial^4 u}{\partial x^4} + \frac{\partial^4 u}{\partial y^4} &= f_{\pi/4} \text{ in } \Omega_{\pi/4} \\ u = \frac{\partial u}{\partial \boldsymbol{n}} &= 0 \text{ on } \Gamma = \partial\Omega_{\pi/4} \end{cases} \tag{3.12}$$

The Python code used to solve this problem is very similar to the previous one given in Listing 3.4: we just have to rotate the function f and the domain. These operations are given in Listing 3.6

```
theta=-pi/4 # For mesh rotation
R=[[cos(theta),-sin(theta)],[sin(theta),cos(theta)]]
P=np.dot([-0.75,0.75],R)
f=lambda x,y:exp(-100*((x - P[0])**2 + (y - P[1])**2))
meshfile=gmsh.buildmesh2d(geoFile,N, force=True,verbose=0)
Th=siMesh(meshfile,trans=lambda q: np.dot(q,R))
```

Listing 3.6: 2D clamped *grid* BVP ...

The complete code is given in file `BVP_ClampedDiagonalGrid_pentagonal01_2D01.py` [6] and can be run with the following python commands:

```
from fc_vfemp1_biharmonic.examples.BVP_ClampedDiagonalGrid_pentagonal01_2D01
    import run
res=run(N=50)
```

[5] directory `fc_vfemp1_biharmonic/examples/`
[6] directory `fc_vfemp1_biharmonic/examples/`

3.3.4 Graphical results

For graphical convenience the numerical solutions on the grid with the diagonal fibers are rotated by $-\pi/4$.

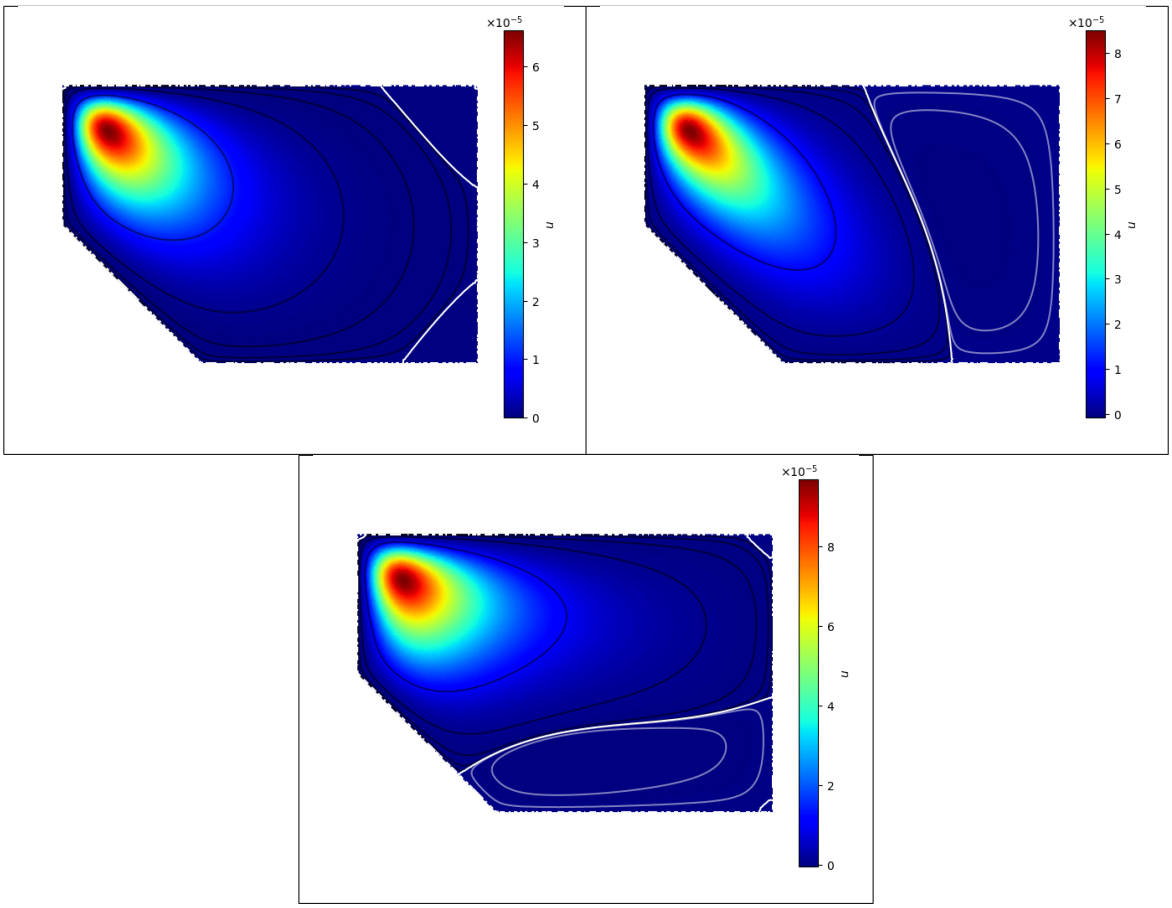


Figure 3.8: Displacement u of, respectively, an isotropic pentagonal plate (upper left), grid with the aligned fibers (upper right) and grid with the diagonal fibers (bottom).

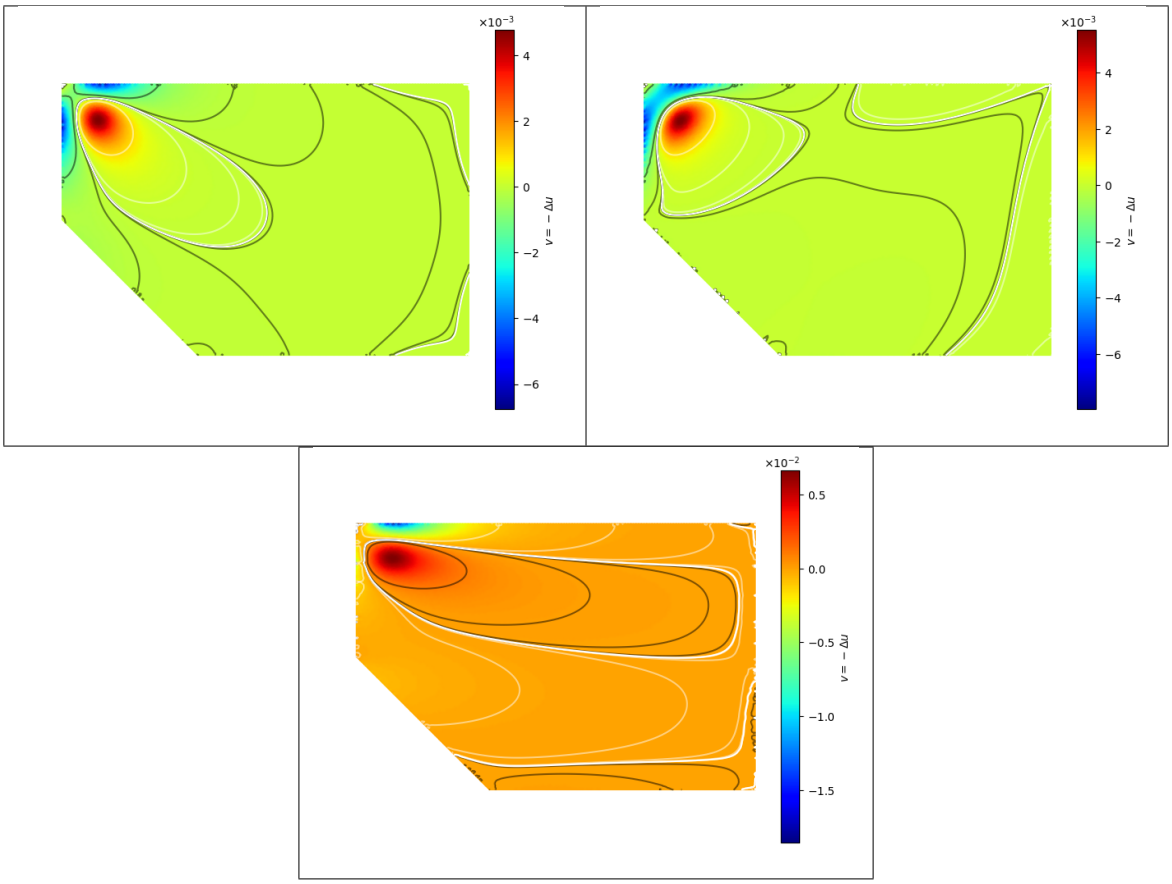


Figure 3.9: $v = -\Delta u$ of, respectively, an isotropic pentagonal plate (upper left), grid with the aligned fibers (upper right) and grid with the diagonal fibers (bottom).

Bibliography

[1] F. Cuvelier. fc_simesh: a object-oriented Python 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.

[2] F. Cuvelier. fc_simesh_matplotlib: an add-on to the fc_simesh Python package for displaying simplices meshes or datas on simplices meshes by using matplotlib python package. `http://www.math.univ-paris13.fr/~cuvelier/software/`, 2017. User's Guide.

[3] F. Cuvelier. fc_simesh_mayavi: an add-on to the fc_simesh Python package for displaying simplices meshes or datas on simplices meshes by using mayavi python package. `http://www.math.univ-paris13.fr/~cuvelier/software/`, 2017. User's Guide.

[4] 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.

[5] 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.

[6] Truong Ha Hai Dang Quang A and Vu Vinh Quang. Iterative method for a biharmonic problem with crack singularities. *Applied Mathematical Sciences*, 6:3095–3108, 2012.

[7] T. Gerasimov. *The clamped elastic grid, a fourth order equation on a domain with corner*. PhD thesis, Delft University of Technology, the Netherlands, 2009.

[8] A. Quarteroni and A. Valli. *Numerical Approximation of Partial Differential Equations*, volume 23 of *Springer Series in Computational Mathematics*. Springer-Verlag, Berlin, 1994.