# Domain Decomposition with ooVecFEM

F. Cuvelier

2016/05/04

## Table des matières

# 1 Problem

Let $\Omega$ be an open bounded subset of $\mathbb{R}^d$, $d \geqslant 1$.

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

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

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 $f \in L^\infty(\Omega)$ 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 [?].

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

A *scalar* boundary value problem is given by

**_Scalar_ Boundary Value Problem**

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

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

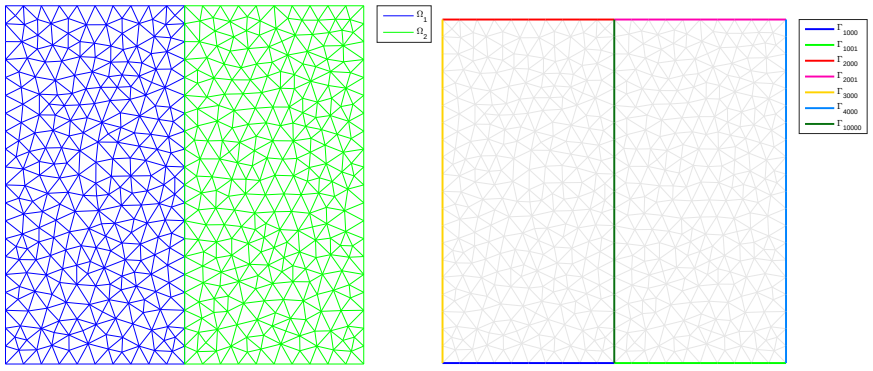
$$u = g^D, \qquad \text{on } \Gamma^D, \tag{1.3}$$

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

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

(1.3) and (1.4) are respectively the Dirichlet, and Robin (or Neumann if $a^R = 0$) boundary conditions.

## 1.1 B.V.P. sample 1




```
meshfile=buildPartRectMesh2d(1,1,2,1,20);
Th=ooMesh(meshfile);
```

Listing 1: Mesh from `sample20.geo`, label of the domains (left) and label of the boundaries (right)

This mesh was generated with gmsh and *RectangleUnifPart01* file by calling under Matlab the function buildPartRectMesh2d (part of mooMesh toolbox):

meshfile=buildPartRectMesh2d(Lx,Ly,Nx,Ny,N);

Let $\Omega = \Omega_1 \cup \Omega_2$, $\Gamma^D = \bigcup_{l \in \mathcal{D}} \Gamma_l$, with $\mathcal{D} = \{1000, 1001, 2000, 2001, 3000, 4000\}$, and $u_{\text{ex}}(x,y) = xy(x-L_x)(x-L_y)$ be the solution of the BVP

**B.V.P. 1**

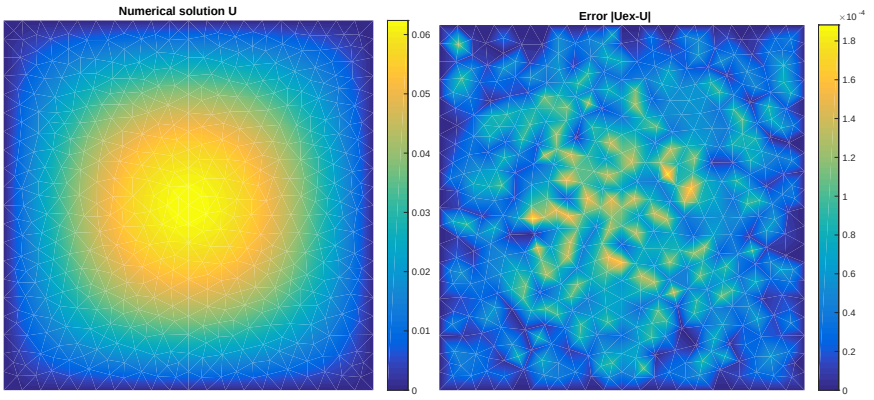
Find $u \in \mathrm{H}^1(\Omega)$ such that

$$\eta u - \Delta u = f, \qquad \text{in } \Omega, \tag{1.5}$$

$$u = 0, \qquad \text{on } \Gamma^D, \tag{1.6}$$

where $\eta > 0$ and $f(x,y) \stackrel{\text{def}}{=} -2(x(x-L_x) + y(y-L_y)) + \eta u_{\text{ex}}(x,y)$.

Without domain decomposition, this problem can be solve using ooVecFEM.




```
Lx=1;Ly=1;
meshfile=buildPartRectMesh2d(Lx,Ly,2,1,20);
Th=ooMesh(meshfile);
uex=@(x,y) (x-Lx).*x.*(y-Ly).*y; % So Dirichlet ==0
eta=1;
f=@(x,y) -2*((x-Lx).*x+(y-Ly).*y) +eta.*uex(x,y);
Lop=Loperator(2,2,{1,0;0,1},[],[],eta);
pde=PDEelement(Lop,f);
bvp=BVP(Th,pde);

idxlab=Th.find(1);
% labD=Th.sThlab(find(Th.sThlab(idxlab)<10000));
labD=[1000,1001,2000,2001,3000,4000];
for lab = labD
  bvp.setDirichlet(lab,uex);
end

u=bvp.solve();
```

Listing 2: Numerical solution (left) and error (right)

## 1.2 B.V.P. sample 2

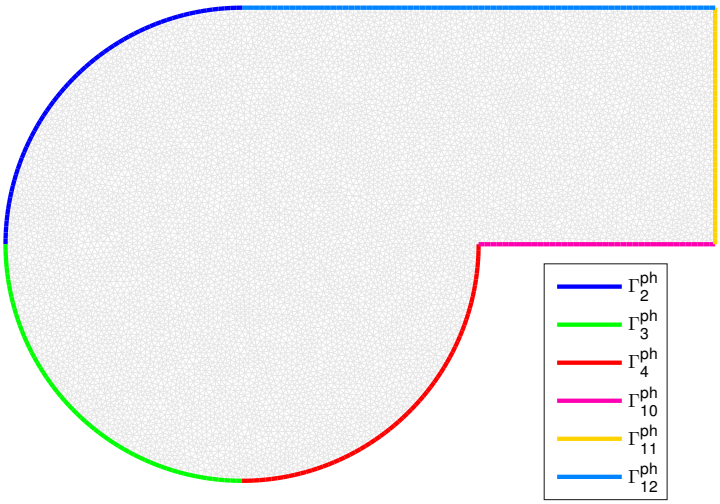



Figure 1: Boundaries

This mesh was generated with gmsh and *sample02.geo* file by calling under Matlab the function BUILDMESH2D (part of mooMesh toolbox):

```
meshfile=buildmesh2d('sample02',20,'force',true,'verbose',0,'check',false);
```

Let $u_{ex}(x,y) = \cos(x-y)\sin(x+y) + e^{-(x^2+y^2)}$ be the solution of the BVP

**B.V.P. 2**

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

$$\eta u - \Delta u = f, \qquad \text{in } \Omega, \tag{1.7}$$

$$u = g^D, \qquad \text{on } \Gamma^D = \Gamma_2^{ph} \cup \Gamma_4^{ph}, \tag{1.8}$$

$$\frac{\partial u}{\partial n} + a^R u = g^R, \qquad \text{on } \Gamma^R = \Gamma_3^{ph} \cup \Gamma_{10}^{ph} \cup \Gamma_{11}^{ph} \cup \Gamma_{12}^{ph}. \tag{1.9}$$

where $\eta > 0$, $a^R = 0$ on $\Gamma_3^{ph} \cup \Gamma_{11}^{ph} \cup \Gamma_{12}^{ph}$ and $a^R(x,y) = 0.1 + e^{x+y}$ on $\Gamma_{10}^{ph}$.
The other functions are computed from $u_{ex}$ and given by $f(x,y) \stackrel{\text{def}}{=} (\eta - \Delta)u_{ex}$, $g^D(x,y) = u_{ex}(x,y)$, $g^R(x,y) = \frac{\partial u_{ex}}{\partial n}(x,y) + a^R(x,y)u_{ex}(x,y)$.

Without domain decomposition, this problem can be solve using ooVecFEM. We give Matlab code

```
meshfile=buildmesh2d('sample02',20,'force',true,'verbose',0,'check',false);
Th=ooMesh(meshfile,'dim',2,'format','gmsh','isPhysLabel',false);
uex=@(x,y) cos(x-y).*sin(x+y)+exp(-(x.^2+y.^2));
eta=1;
aR=@(x,y) 0.1+exp(x+y);
...
Lop=Loperator(dim,d,{1,0;0,1},[],[],eta);
pde=PDEelement(Lop,f);
bvp=BVP(Th,pde);
for lab=[2,4]
  bvp.setDirichlet(lab,uex);
end
bvp.setRobin(10,@(x,y) -gradu{2}(x,y)+aR(x,y).*uex(x,y),aR);
bvp.setRobin(12,@(x,y) gradu{2}(x,y),0);
bvp.setRobin(11,@(x,y) gradu{1}(x,y),0);
bvp.setRobin(3,@(x,y) x.*gradu{1}(x,y)+y.*gradu{2}(x,y),0);
u=bvp.solve();
Th.plotVal(u);
```

# 2 Domain Decomposition using optimized Schwarz method

Let $\Omega = \bigcup_{i=1}^{M} \Omega_i$ such that $\Omega_i$, $1 \leqslant i \leqslant M$ are convex polygons ($d = 2$) or polyhedra ($d = 3$). We assume also that this decomposition is geometrically conforming in the sense that the intersection of the closure of two different subdomains, if not empty, is either a common vertex or a common edge.

For all $i \in [\![1, M]\!]$, we note

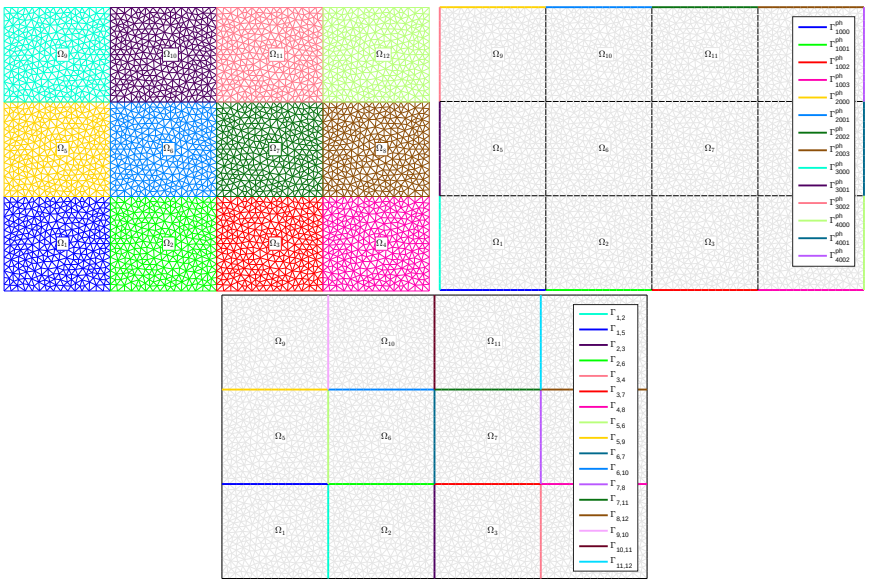
$$\Gamma_i^D = \partial\Omega_i \cap \Gamma^D, \quad \Gamma_i^N = \partial\Omega_i \cap \Gamma^N, \quad \Gamma_i^R = \partial\Omega_i \cap \Gamma^R \tag{2.1}$$

and

$$\Gamma_{i,j} = \begin{cases} \partial\Omega_i \cap \partial\Omega_j, & \forall j \in [\![1, M]\!] \backslash \{i\}, \\ \varnothing, & \text{if } i = j. \end{cases} \tag{2.2}$$

Note that $\Gamma_{i,j} = \Gamma_{j,i}$.

In Listing 4 we represent the uniformly partionned mesh of the rectangle $[0, 3] \times [0, 2]$ with $4 - by - 3$ partitions. This mesh was generated via gmsh by an explicit build of all the partitions in the *.geo* file `RectangleUnifPart01.geo`.



```
1 Lx=3;Ly=2;Nx=4;Ny=3;
2 meshfile=buildPartRectMesh2d(Lx,Ly,Nx,Ny,20);
3 Th=ooMeshDD(ooMesh(meshfile));
```

Listing 3: Mesh from `RectangleUnifPart01.geo`, labels of the partitions (upper left), labels of the boundaries (upper right) and labels of interfaces (bottom)

In Listing 2 we represent the partionned mesh of a domain with seven partitions using gmsh with Metis.

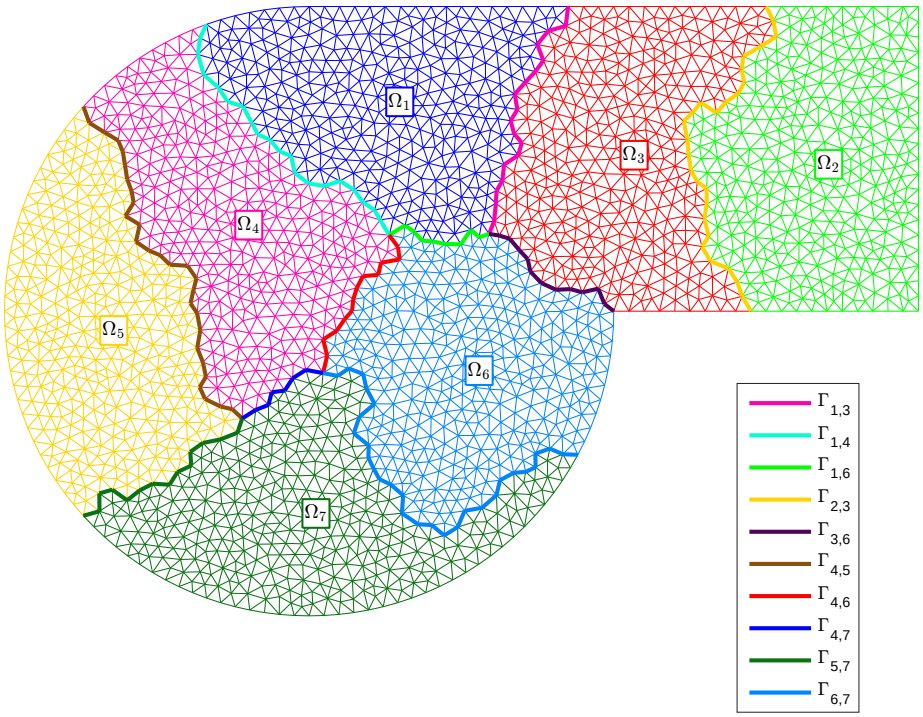



Figure 2: Partition sample with interfaces

Let $i \in [\![1, M]\!]$, we note

$$\Gamma_i^{\text{int}} = \bigcup_{j=1}^{M} \Gamma_{i,j}, \qquad \Gamma^{\text{int}} = \bigcup_{i=1}^{M} \Gamma_i^{\text{int}}, \tag{2.3}$$

and

$$\mathrm{L}^p(\Gamma_i^{\text{int}}) := \prod_{j=1}^{M} \mathrm{L}^p(\Gamma_{i,j}), \quad \text{and } \mathrm{L}^p(\Gamma^{\text{int}}) := \prod_{i=1}^{M} \mathrm{L}^p(\Gamma_i^{\text{int}}), \quad 1 \leqslant p \leqslant \infty, \tag{2.4}$$

with the convention that $L^p(\varnothing) = \varnothing$.
Let $\boldsymbol{g} \in \mathrm{L}^p(\Gamma^{\text{int}})$ then we note

$$\boldsymbol{g} = (\boldsymbol{g}_1, \cdots, \boldsymbol{g}_M) \in \mathrm{L}^p(\Gamma_1^{\text{int}}) \times \cdots \mathrm{L}^p(\Gamma_M^{\text{int}}) \tag{2.5}$$

where $\boldsymbol{g}_i = (g_{i,1}, \cdots, g_{i,M}) \in \mathrm{L}^p(\Gamma_i^{\text{int}})$ or with matricial notations

$$\boldsymbol{g} = \begin{pmatrix} g_{1,1} & \cdots & g_{1,M} \\ \vdots & \ddots & \vdots \\ g_{M,1} & \cdots & g_{M,M} \end{pmatrix} \text{ where } g_{i,j} \in \mathrm{L}^p(\Gamma_{i,j}) \tag{2.6}$$

With these notations, we can note that $g_{i,i} = \varnothing$, $\forall i \in [\![1, M]\!]$.

We have $\partial\Omega_i = \Gamma_i^D \cup \Gamma_i^R \cup \Gamma_i$.

The coefficients in $\mathcal{L}$ may be discontinuous and we denote by $\mathcal{L}_i$ (resp. $\mathbb{A}_i$, $\boldsymbol{b}_i$) the restriction of $\mathcal{L}$ (resp. $\mathbb{A}$, $\boldsymbol{b}$) to $\Omega_i$. A vérifier !!! Discontinuité sur l'interface???

Let $\boldsymbol{n}_i$ be the unit outward pointing vector field is on $\partial\Omega_i$. With sufficient regularity, on can prove that solving problem (1.2)-(1.4) is equivalent to solve the subdomain problems,

**Boundary Value Problem on all $\Omega_i$**

For $i \in [\![1, M]\!]$, find $u_i \in \mathrm{H}^1(\Omega_i)$ such that

$$\mathcal{L}(u_i) = f, \qquad \text{in } \Omega_i, \tag{2.7}$$

$$u_i = g^D, \qquad \text{on } \Gamma_i^D, \tag{2.8}$$

$$\frac{\partial u_i}{\partial n^i_{\mathcal{L}_i}} + a^R u_i = g^R, \qquad \text{on } \Gamma_i^R, \tag{2.9}$$

with $\frac{\partial u}{\partial n_{\mathcal{L}_i}} = \langle \mathbb{A}_i \boldsymbol{\nabla} u, \boldsymbol{n}_i \rangle - \langle \boldsymbol{b}_i u, \boldsymbol{n}_i \rangle$ on $\partial\Omega_i$, together with the physical transmission conditions :

$$u_i = u_j, \tag{2.10}$$

$$\frac{\partial u_i}{\partial n_{\mathcal{L}_i}} = -\frac{\partial u_j}{\partial n_{\mathcal{L}_j}} \qquad \text{on } \Gamma_{i,j}, \ \forall j \in [\![1, M]\!]. \tag{2.11}$$

Note that on $\Gamma_{i,j}$, using $\boldsymbol{n}_j = -\boldsymbol{n}_i$ we have $-\dfrac{\partial u_j}{\partial n_{\mathcal{L}_i}} = \dfrac{\partial u_j}{\partial n_{\mathcal{L}_j}}$. When there is no ambiguity one denotes $\frac{\partial u_j}{\partial n_{\mathcal{L}_j}} := \frac{\partial u_j}{\partial n_{\mathcal{L}}}$. The condition (2.10) comes from the $H^1$ regularity while the condition (2.11) comes from the weak formulation.

Let $\boldsymbol{u} = (u_1, \ldots, u_M) \in \bigotimes_{i=1}^{M} \mathrm{H}^1(\Omega_i)$.

Let $\alpha_{i,j}$ be positive functions in $L^\infty(\Gamma_{i,j})$. One can see that on each none empty $\Gamma_{i,j}$, the physical transmission conditions (2.10)-(2.11) are equivalent to the Robin transmission conditions

$$\frac{\partial u_i}{\partial n_{\mathcal{L}_i}} + \alpha_{i,j} u_i = -\frac{\partial u_j}{\partial n_{\mathcal{L}_j}} + \alpha_{i,j} u_j, \quad \text{on } \Gamma_{i,j}, \tag{2.12}$$

$$\frac{\partial u_j}{\partial n_{\mathcal{L}_j}} + \alpha_{j,i} u_j = -\frac{\partial u_i}{\partial n_{\mathcal{L}_j}} + \alpha_{j,i} u_i, \quad \text{on } \Gamma_{i,j}. \tag{2.13}$$

Indeed, if we substract (2.13) from (2.12) we obtain $(\alpha_{j,i} + \alpha_{i,j})(u_i - u_j) = 0$, and thus $u_i = u_j$ on $\Gamma_{i,j}$. Then, using $u_i = u_j$ in (2.12) we obtain $\frac{\partial u_i}{\partial n_{\mathcal{L}_i}} = -\frac{\partial u_j}{\partial n_{\mathcal{L}_j}}$ on $\Gamma_{i,j}$. Thus, solving (1.2)-(1.4) is equivalent to solve the Robin problems

**Robin Boundary Value Problem on $\Omega_i$**

For $i \in [\![1, M]\!]$, find $u_i \in \mathrm{H}^2(\Omega_i)$ such that

$$\mathcal{L}(u_i) = f, \qquad \text{in } \Omega_i, \tag{2.14}$$

$$u_i = g^D, \qquad \text{on } \Gamma_i^D, \tag{2.15}$$

$$\frac{\partial u_i}{\partial n_{\mathcal{L}_i}} + a^R u_i = g^R, \qquad \text{on } \Gamma_i^R, \tag{2.16}$$

with $\frac{\partial u}{\partial n_{\mathcal{L}_i}} = \langle \mathbb{A}_i \boldsymbol{\nabla} u, \boldsymbol{n}_i \rangle - \langle \boldsymbol{b}_i u, \boldsymbol{n}_i \rangle$ on $\partial\Omega_i$,
together with the Robin transmission conditions :

$$\frac{\partial u_i}{\partial n_{\mathcal{L}_i}} + \alpha_{i,j} u_i = -\frac{\partial u_j}{\partial n_{\mathcal{L}_j}} + \alpha_{i,j} u_j, \qquad \text{on } \Gamma_{i,j}, \ \ \forall j \in [\![1, M]\!]. \tag{2.17}$$

We note $\xi_{i,j}$ the contribution of domain $\Omega_j$ to domain $\Omega_i$ (right hand side of (2.17)) defined by

$$\xi_{i,j} := -\frac{\partial u_j}{\partial n_{\mathcal{L}_j}} + \alpha_{i,j} u_j \tag{2.18}$$

We have

$$\xi_{i,j} := -\frac{\partial u_j}{\partial n_{\mathcal{L}_j}} + \alpha_{i,j} u_j = -(\frac{\partial u_j}{\partial n_{\mathcal{L}_j}} + \alpha_{j,i} u_j) + (\alpha_{j,i} + \alpha_{i,j}) u_j$$

From (2.17) on $\Gamma_{j,i}$ we have

$$\frac{\partial u_j}{\partial n_{\mathcal{L}_j}} + \alpha_{j,i} u_j = -\frac{\partial u_i}{\partial n_{\mathcal{L}_i}} + \alpha_{j,i} u_i := \xi_{j,i}$$

and then

$$\xi_{i,j} = -\xi_{j,i} + (\alpha_{j,i} + \alpha_{i,j}) u_j \tag{2.19}$$

In the next section we will show that solving of problems (2.14)-(2.17), for $i \in [\![1, M]\!]$, is equivalent to solve an interface problem (i.e. where the unknowns are located only on the interfaces, with two unknowns on each interface $\Gamma_{i,j}$ : $\frac{\partial u_i}{\partial n_{\mathcal{L}_i}} + \alpha_{i,j} u_i$ and $\frac{\partial u_j}{\partial n_{\mathcal{L}_j}} + \alpha_{j,i} u_j$).

## 2.1 Model problem

**Boundary Value Problem model 1 on all $\Omega_i$**

For $i \in [\![1, M]\!]$, find $u_i \in \mathrm{H}^1(\Omega_i)$ such that

$$\eta u_i - \Delta u_i = f, \qquad \text{in } \Omega_i, \tag{2.20}$$

$$u_i = g^D, \qquad \text{on } \Gamma_i^D, \tag{2.21}$$

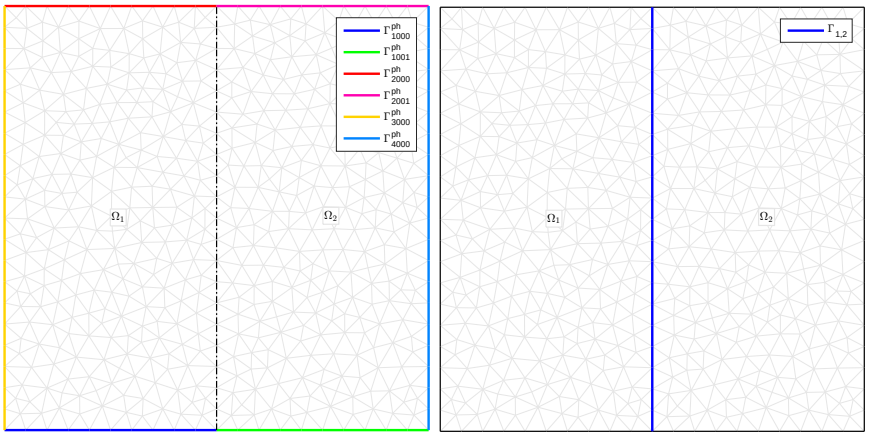
$$\frac{\partial u_i}{\partial n_i} + a^R u_i = g^R, \qquad \text{on } \Gamma_i^R, \tag{2.22}$$

with the Robin transmission conditions :

$$\frac{\partial u_i}{\partial n_i} + \alpha_{i,j} u_i = -\frac{\partial u_j}{\partial n_j} + \alpha_{i,j} u_j, \qquad \text{on } \Gamma_{i,j}, \ \forall j \in [\![1, M]\!]. \tag{2.23}$$

### 2.1.1 BVP sample 1

For this sample, the partionned mesh is given in Listing 4.




```
Lx=1;Ly=1;Nx=2;Ny=1;
meshfile=buildPartRectMesh2d(Lx,Ly,Nx,Ny,20);
Th=ooMeshDD(ooMesh(meshfile));
```

Listing 4: Mesh from `RectangleUnifPart01.geo`, labels of the partitions and boundaries (left) and label of the interface (right)

**Boundary Value Problem sample 1 on $\Omega = \Omega_1 \cup \Omega_2$**

find $(u_1, u_2) \in \mathrm{H}^1(\Omega_1) \times \mathrm{H}^1(\Omega_2)$ such that, forall $i \in \{1, 2\}$,

$$\eta u_i - \Delta u_i = f, \qquad \text{in } \Omega_i, \tag{2.24}$$

$$u_i = 0, \qquad \text{on } \partial\Omega, \tag{2.25}$$

with the Robin transmission conditions :

$$\frac{\partial u_1}{\partial n_1} + \alpha_{1,2} u_1 = -\frac{\partial u_2}{\partial n_2} + \alpha_{1,2} u_2, \qquad \text{on } \Gamma_{1,2}, \tag{2.26}$$

$$\frac{\partial u_2}{\partial n_2} + \alpha_{2,1} u_2 = -\frac{\partial u_1}{\partial n_1} + \alpha_{2,1} u_2, \qquad \text{on } \Gamma_{2,1} = \Gamma_{1,2}, \tag{2.27}$$

To set these two B.V.P. without the Robin transmission conditions under Matlab using mooVecFem, one can used the following commands :

```
Th=ooMeshDD(partmeshfile);
...
Lop=Loperator(dim,d,{1,0;0,1},[],[],eta);
pde=PDEelement(Lop,f);
DDbvp=DD(Th,pde);
for lab=[1000,1001,2000,2001,3000,4000]
  DDbvp.setDirichlet(lab,0);
end
```

- DDbvp=DD(Th,pde); : on each partition, set (2.24) using `pde`.
- DDbvp.setDirichlet(lab,uex); : on each partition, on boundary with label `lab` (if exists) set the Dirichlet boundary condition (2.25).

### 2.1.2 BVP sample 2

**Boundary Value Problem sample 1 on all $\Omega_i$**

For $i \in [\![1, M]\!]$, find $u_i \in \mathrm{H}^1(\Omega_i)$ such that

$$\eta u_i - \Delta u_i = f, \qquad \text{in } \Omega_i, \tag{2.28}$$

$$u_i = g^D, \qquad \text{on } \Gamma_i^D, \tag{2.29}$$

$$\frac{\partial u_i}{\partial n_i} + a^R u_i = g^R, \qquad \text{on } \Gamma_i^R, \tag{2.30}$$

with the Robin transmission conditions :

$$\frac{\partial u_i}{\partial n_i} + \alpha_{i,j} u_i = -\frac{\partial u_j}{\partial n_j} + \alpha_{i,j} u_j, \qquad \text{on } \Gamma_{i,j}, \ \forall j \in [\![1, M]\!]. \tag{2.31}$$

To set these $M$ B.V.P. without the Robin transmission conditions under Matlab using mooVecFem, one can used the following commands :

```
Th=ooMeshDD(partmeshfile,'dim',2,'format','gmsh');
...
Lop=Loperator(dim,d,{1,0;0,1},[],[],eta);
```

```
pde=PDEelement(Lop,f);
DDbvp=DD(Th,pde);
for lab=[2,4]
  DDbvp.setDirichlet(lab,uex);
end
DDbvp.setRobin(10,@(x,y) -gradu{2}(x,y)+aR(x,y).*uex(x,y),aR);
DDbvp.setRobin(12,@(x,y) gradu{2}(x,y),0);
DDbvp.setRobin(11,@(x,y) gradu{1}(x,y),0);
DDbvp.setRobin(3,@(x,y) x.*gradu{1}(x,y)+y.*gradu{2}(x,y),0);
```

- DDbvp=DD(Th,pde); : on each partition, set (2.28) using `pde`.
- DDbvp.setDirichlet(lab,uex); : on each partition, on boundary with label `lab` (if exists) set the Dirichlet boundary condition (2.29).
- DDbvp.setRobin(lab,g, aR); : on each partition, on boundary with label `lab` (if exists) set the Robin boundary condition (2.30).

## 2.2 Interface problems

### 2.2.1 Notations

We first introduce some notations : let $i \in \{1, ..., M\}$, $\mathcal{V}_i = L^2(\Omega_i) \times H^{\frac{1}{2}}(\Gamma_i^D) \times L^2(\Gamma_i^R)$, and $\mathcal{V} = \prod_{i=1}^M \mathcal{V}_i$. We note $\boldsymbol{\alpha}_i \in L^\infty(\Gamma_i)$ the function defined on $\Gamma_i$ such that $\forall j \in [\![1, M]\!]$, $\boldsymbol{\alpha}_i = \alpha_{i,j}$ on $\Gamma_{i,j}$, and $\boldsymbol{\alpha} = (\alpha_1, ..., \alpha_M)$. Let $\mathrm{H}^1_*(\Omega_i) = \{v \in H^1(\Omega_i),\ v = 0 \text{ on } \Gamma_i^D\}$. We introduce the subproblem solution operator for $\Omega_i$, as

**$\mathcal{M}_i$ : subproblem solution operator for $\Omega_i$**

Let $\boldsymbol{\zeta}_i = (\zeta_{i,1}, ..., \zeta_{i,M}) \in \mathrm{L}^2(\Gamma_i)$ and $\boldsymbol{\mathcal{F}}_i = (f_i, g_i^D, g_i^R) \in \mathcal{V}_i$. Let $w_i \in \mathrm{H}^1(\Omega_i)$ the solution of

$$\begin{cases} \mathcal{L}_i(w) = f_i, & \text{in } \Omega_i & \text{(2.32a)} \\ w = g_i^D, & \text{on } \Gamma_i^D, & \text{(2.32b)} \\ \dfrac{\partial w}{\partial n^i_{\mathcal{L}_i}} + a_i^R w = g_i^R, & \text{on } \Gamma_i^R, & \text{(2.32c)} \\ \dfrac{\partial w}{\partial n^i_{\mathcal{L}_i}} + \alpha_{i,j} w = \zeta_{i,j}, & \text{on } \Gamma_{i,j},\ j \in [\![1, M]\!]. & \text{(2.32d)} \end{cases}$$

The subproblem solution operator $\mathcal{M}_i$ is defined by

$$\begin{array}{rccc} \mathcal{M}_i & : & L^2(\Gamma_i) \times \mathcal{V}_i & \longrightarrow & \mathrm{H}^1(\Omega_i) \\ & & (\boldsymbol{\zeta}_i, \boldsymbol{\mathcal{F}}_i) & \longmapsto & w_i \end{array} \qquad \text{(2.33)}$$

We can note that if $\boldsymbol{\zeta}_i$ is known, one can localy solve (2.32). The operator $\mathcal{M}_i$ also depends on $a_i^R$ and $(\alpha_{i,j})_{j=1}^M$ but to reduce the notations we ommited this dependance.

The following lemma is right a consequence of the linearity of the operators $\mathcal{L}_i$

**Lemme 2.1**

Let $(\boldsymbol{\zeta}_i, \boldsymbol{\mathcal{F}}_i) \in L^2(\Gamma_i) \times \mathcal{V}_i$, then we have

$$\mathcal{M}_i(\boldsymbol{\zeta}_i, \boldsymbol{\mathcal{F}}_i) = \mathcal{M}_i(0, \boldsymbol{\mathcal{F}}_i) + \mathcal{M}_i(\boldsymbol{\zeta}_i, 0) \tag{2.34}$$

We now introduce the problem solution operator $\mathcal{M}$ on $\Omega_1 \times \cdots \times \Omega_M$ as

**$\mathcal{M}$ : problem solution operator on $\Omega_1 \times \cdots \times \Omega_M$**

Let $\boldsymbol{\zeta} = (\boldsymbol{\zeta}_1, \ldots, \boldsymbol{\zeta}_M) = (\zeta_{i,j})_{i,j=1}^M \in \mathrm{L}^2(\Gamma^{\mathrm{int}})$ where $\zeta_{i,j} \in \mathrm{L}^2(\Gamma_{i,j})$. Let $\boldsymbol{\mathcal{F}} = (f, g^D, g^R) \in \mathcal{V}$ and $\boldsymbol{\mathcal{F}}_i = (f_i, g_i^D, g_i^R) \in \mathcal{V}_i$ such that $f_i = f_{|\Omega_i}$, $g_i^D = g^D_{|\Gamma_i^D}$, and $g_i^R = g^R_{|\Gamma_i^R}$.
Let $(w_1, \ldots, w_M) = (\mathcal{M}_1(\boldsymbol{\zeta}_1, \boldsymbol{\mathcal{F}}_1), \ldots, \mathcal{M}_M(\boldsymbol{\zeta}_M, \boldsymbol{\mathcal{F}}_M))$ be the subproblem solutions on $\Omega_1 \times \cdots \times \Omega_M$. We define the operator $\mathcal{M}$ as

$$\begin{array}{rccc} \mathcal{M} & : & \mathrm{L}^2(\Gamma^{\mathrm{int}}) \times \mathcal{V} & \longrightarrow & \mathrm{H}^1(\Omega_1) \times \ldots \times \mathrm{H}^1(\Omega_M) \\ & & (\boldsymbol{\zeta}, \boldsymbol{\mathcal{F}}) & \longmapsto & (w_1, \ldots, w_M) \end{array} \tag{2.35}$$

From previous lemma we obtain

**Lemme 2.2**

Let $(\boldsymbol{\zeta}, \boldsymbol{\mathcal{F}}) \in \mathrm{L}^2(\Gamma^{\mathrm{int}}) \times \mathcal{V}$, then we have

$$\mathcal{M}(\boldsymbol{\zeta}, \boldsymbol{\mathcal{F}}) = \mathcal{M}(0, \boldsymbol{\mathcal{F}}) + \mathcal{M}(\boldsymbol{\zeta}, 0) \tag{2.36}$$

We now introduce the trace operator $\mathcal{B}$ on interfaces $\Gamma^{int}$

**$\mathcal{B}$ : trace operator on interfaces $\Gamma^{int}$**

$$\begin{array}{rccc} \mathcal{B} & : & \mathrm{H}^1(\Omega_1) \times \cdots \times \mathrm{H}^1(\Omega_M) & \longrightarrow & \mathrm{L}^2(\Gamma^{\mathrm{int}}) \\ & & (w_1, \ldots, w_M) & \longmapsto & (\mathcal{B}_1(w_1), \ldots, \mathcal{B}_M(w_M))^t \end{array} \tag{2.37}$$

where, for all $i \in [\![1, M]\!]$,

$$\begin{array}{rccc} \mathcal{B}_i & : & \mathrm{H}^1(\Omega_i) & \longrightarrow & L^2(\Gamma_i) = L^2(\Gamma_{i,1}) \times \cdots \times L^2(\Gamma_{i,M}) \\ & & w_i & \longmapsto & (\mathcal{B}_{i,1}(w_i), \ldots, \mathcal{B}_{i,M}(w_i)) \end{array} \tag{2.38}$$

with, for all $j \in [\![1, M]\!]$,

$$\begin{array}{rccc} \mathcal{B}_{i,j} & : & \mathrm{H}^1(\Omega_i) & \longrightarrow & L^2(\Gamma_{i,j}) \\ & & w_i & \longmapsto & -\frac{\partial w_i}{\partial n_{\mathcal{L}_i}} + \alpha_{j,i} w_i \end{array} \tag{2.39}$$

So we have with matricial notations

$$\mathcal{B}(w_1, \ldots, w_M) = \begin{pmatrix} -\frac{\partial w_1}{\partial n_{\mathcal{L}_1}} + \alpha_{1,1} w_1 & \cdots & -\frac{\partial w_1}{\partial n_{\mathcal{L}_1}} + \alpha_{M,1} w_1 \\ \vdots & \ddots & \vdots \\ -\frac{\partial w_M}{\partial n_{\mathcal{L}_M}} + \alpha_{1,M} w_M & \cdots & -\frac{\partial w_M}{\partial n_{\mathcal{L}_M}} + \alpha_{M,M} w_M \end{pmatrix} \tag{2.40}$$

# 3 Fixed point problems

Let $\boldsymbol{u} = (u_1, \ldots, u_M)$ be the solutions of (2.14)-(2.17) and $\boldsymbol{\xi} = (\xi_{i,j})_{i,j=1}^{M} \in \mathrm{L}^2(\Gamma^{\mathrm{int}})$ where $\xi_{i,j}$ is defined in (2.18) by

$$\xi_{i,j} := -\frac{\partial u_j}{\partial n_{\mathcal{L}_j}} + \alpha_{i,j} u_j.$$

Using previous operators we have

$$\boldsymbol{u} = \mathcal{M}(\boldsymbol{\xi}, \boldsymbol{\mathcal{F}})$$

and

$$\begin{aligned} \mathcal{B}(\boldsymbol{u}) &= \begin{pmatrix} -\frac{\partial u_1}{\partial n_{\mathcal{L}}} + \alpha_{1,1} u_1 & \cdots & -\frac{\partial u_1}{\partial n_{\mathcal{L}}} + \alpha_{M,1} u_1 \\ \vdots & \ddots & \vdots \\ -\frac{\partial u_M}{\partial n_{\mathcal{L}}} + \alpha_{1,M} u_M & \cdots & -\frac{\partial u_M}{\partial n_{\mathcal{L}}} + \alpha_{M,M} u_M \end{pmatrix} \\ &= \begin{pmatrix} \xi_{1,1} & \cdots & \xi_{M,1} \\ \vdots & \ddots & \vdots \\ \xi_{1,M} & \cdots & \xi_{M,M} \end{pmatrix} = \boldsymbol{\xi}^{\mathsf{t}}. \end{aligned}$$

## 3.1 Partitoned problem

With previous notations, we obtain the **partitioned problem**

$$\boldsymbol{u} = \mathcal{M}(\mathcal{B}(\boldsymbol{u})^{\mathsf{t}}, \boldsymbol{\mathcal{F}}) \tag{3.1}$$

**Fixed point partitioned problem**

Let $\mathcal{P}$ be the partitioned operator defined by

$$\begin{array}{rccc} \mathcal{P} & : & \bigotimes_{i=1}^{M} \mathrm{H}^1(\Omega_i) \times \mathcal{V} & \longrightarrow & \bigotimes_{i=1}^{M} \mathrm{H}^1(\Omega_i) \\ & & (\boldsymbol{u}, \boldsymbol{\mathcal{F}}) & \longmapsto & \mathcal{M}(\mathcal{B}(\boldsymbol{u})^{\mathsf{t}}, \boldsymbol{\mathcal{F}}) \end{array} \tag{3.2}$$

Let $\boldsymbol{\mathcal{F}} = (f, g^D, g^R) \in \mathcal{V}$. The partitioned unknow $\boldsymbol{u} = (u_1, \ldots, u_M) \in \bigotimes_{i=1}^{M} \mathrm{H}^1(\Omega_i)$ is solution of the fixed point partitioned problem :

$$\text{Find } \boldsymbol{u} \in \bigotimes_{i=1}^{M} \mathrm{H}^1(\Omega_i), \text{ such that } \boldsymbol{u} = \mathcal{P}(\boldsymbol{u}, \boldsymbol{\mathcal{F}}). \tag{3.3}$$

To solve the fixed point problem (3.3), one can used the **fixed point algorithm**

$$\boldsymbol{u}^{[n+1]} = \mathcal{P}(\boldsymbol{u}^{[n]}, \boldsymbol{\mathcal{F}}), \ \forall n \geqslant 0 \tag{3.4}$$

with a given $\boldsymbol{u}^{[0]} \in \bigotimes_{i=1}^{M} \mathrm{H}^1(\Omega_i)$.

## 3.2 Interfaces problem

With previous notations, we have the **interfaces problem**

$$\boldsymbol{\xi}^{\mathsf{t}} = \mathcal{B} \circ \mathcal{M}(\boldsymbol{\xi}, \boldsymbol{\mathcal{F}}) \quad \text{or} \quad \boldsymbol{\xi} = (\mathcal{B} \circ \mathcal{M}(\boldsymbol{\xi}, \boldsymbol{\mathcal{F}}))^{\mathsf{t}} \tag{3.5}$$

**Interace problem with interface operator**

Let $\mathcal{S}$ be the interface operator defined by

$$\begin{array}{rccc} \mathcal{S} & : & \mathrm{L}^2(\Gamma^{\mathrm{int}}) \times \mathcal{V} & \longrightarrow & \mathrm{L}^2(\Gamma^{\mathrm{int}}) \\ & & (\boldsymbol{\xi}, \boldsymbol{\mathcal{F}}) & \longmapsto & (\mathcal{B} \circ \mathcal{M}(\boldsymbol{\xi}, \boldsymbol{\mathcal{F}}))^{\mathrm{t}} \end{array} \tag{3.6}$$

Let $\boldsymbol{\mathcal{F}} = (f, g^D, g^R) \in \mathcal{V}$. The interface unknow $\boldsymbol{\xi} = (\xi_{i,j})_{i,j=1}^{M} \in \mathrm{L}^2(\Gamma^{\mathrm{int}})$ with $\xi_{i,j}$ defined in (2.18) is solution of the fixed point interface problem :

$$\text{Find } \boldsymbol{\xi} \in \mathrm{L}^2(\Gamma^{\mathrm{int}}), \text{ such that } \boldsymbol{\xi} = \mathcal{S}(\boldsymbol{\xi}, \boldsymbol{\mathcal{F}}). \tag{3.7}$$

To solve the fixed point problem (3.7), one can used the **fixed point algorithm**

$$\boldsymbol{\xi}^{[n+1]} = \mathcal{S}(\boldsymbol{\xi}^{[n]}, \boldsymbol{\mathcal{F}}), \ \forall n \geqslant 0 \tag{3.8}$$

with a given $\boldsymbol{\xi}^{[0]} \in \mathrm{L}^2(\Gamma^{\mathrm{int}})$. This method is detailled in section 5.

An other way is to note that by linearity of operators the fixed point interface problem (3.7) can be rewritten as

$$\boldsymbol{\xi} = \mathcal{S}(\boldsymbol{\xi}, 0) + \mathcal{S}(0, \boldsymbol{\mathcal{F}})$$

and so with $\mathcal{A}(\boldsymbol{\xi}) \stackrel{\text{def}}{=} \boldsymbol{\xi} - \mathcal{S}(\boldsymbol{\xi}, 0)$ and $\boldsymbol{b} = \mathcal{S}(0, \boldsymbol{\mathcal{F}})$ we obtain

$$\mathcal{A}(\boldsymbol{\xi}) = \boldsymbol{b}$$

One can solve this equation using Krylov methods and this is the object of section 6.

But first of all we will give some results from [],... to optimize choices of all $\alpha_{i,j}$ parameters.

# 4 Choices for Robin parameters

## 4.1 Constant parameters

We choose $\alpha_{i,j} = C > 0$. The Matlab function RobinParamCst which computes all $\alpha_{i,j}$ is given in section A.1, Listing 12.

## 4.2 Optimal Robin parameters of order 0

On not empty interface $\Gamma_{i,j}$, we choose $\alpha_{i,j} = \alpha_{j,i} = p^*$ where $p^*$ is given by formula (4.15) of [1] :

$$p^* = \left((k_{\min}^2 + \eta)(k_{\max}^2 + \eta)\right)^{\frac{1}{4}} \tag{4.1}$$

with $k_{\min} = \pi/L$ ($L$ length of $\Gamma_{i,j}^h$) and $k_{\max} = \pi/h$ ($h$ is the maximum of the length edges of $\Gamma_{i,j}^h$).

The Matlab function OptRobinParamOrder0 which computes all $\alpha_{i,j}$ is given in section A.1, Listing 24.

## 4.3 Two-sided optimal Robin parameters

On an not empty interface $\Gamma_{i,j}$, we choose $\alpha_{i,j} = p_1^*$ and $\alpha_{j,i} = p_2^*$ given by formula (4.42) of [1] :

$$p_1^* = \frac{1 - \sqrt{1 + 4\eta(q^*)^2 - 4p^*q^*}}{2q^*}, \quad p_2^* = \frac{1 + \sqrt{1 + 4\eta(q^*)^2 - 4p^*q^*}}{2q^*}, \tag{4.2}$$

where $p^*$ and $q^*$ are given by formula (4.30) of [1] :

$$p^* = \frac{k_{\max}^2\sqrt{k_{\min}^2+\eta} - k_{\min}^2\sqrt{k_{\max}^2+\eta}}{\sqrt{2(k_{\max}^2-k_{\min}^2)}\left((\sqrt{k_{\max}^2+\eta}-\sqrt{k_{\min}^2+\eta})((k_{\max}^2+\eta)\sqrt{k_{\min}^2+\eta}-(k_{\min}^2+\eta)\sqrt{k_{\max}^2+\eta})\right)^{\frac{1}{4}}}$$

$$q^* = \frac{(\sqrt{k_{\max}^2+\eta}-\sqrt{k_{\min}^2+\eta})^{\frac{3}{4}}}{\sqrt{2(k_{\max}^2-k_{\min}^2)}\left((k_{\max}^2+\eta)\sqrt{k_{\min}^2+\eta}-(k_{\min}^2+\eta)\sqrt{k_{\max}^2+\eta})\right)^{\frac{1}{4}}}$$

with $k_{\min} = \pi/L$ ($L$ length of $\Gamma_{i,j}^h$) and $k_{\max} = \pi/h$ ($h$ is the maximum of the length edges of $\Gamma_{i,j}^h$).

The Matlab function OptRobinParamTwoSided which computes all $\alpha_{i,j}$ is given in section A.1, Listing 40.

# 5 Fixed point algorithm

To obtain $\boldsymbol{\xi}^{[n+1]}$ given by the fixed point algorithm (3.8) one can successively compute :

1. $\boldsymbol{u}^{[n+1]} = \mathcal{M}(\boldsymbol{\xi}^{[n]}, \boldsymbol{\mathcal{F}})$
2. and then $\boldsymbol{\xi}^{[n+1]} = \mathcal{B}(\boldsymbol{u}^{[n+1]})^{\mathsf{t}}$.

The computation of $\boldsymbol{u}^{[n+1]} = (\boldsymbol{u}_1^{[n+1]}, \ldots, \boldsymbol{u}_M^{[n+1]})$ is done by solving independently on each partition the local problems :

$$u_i{}^{[n+1]} = \mathcal{M}_i(\xi_i{}^{[n]}, \boldsymbol{\mathcal{F}}_i), \ \forall i \in [\![1, M]\!]. \tag{5.1}$$

For the second step, we can note that

$$\xi_{i,j}^{[n+1]} := -\frac{\partial u_j{}^{[n+1]}}{\partial n_{\mathcal{L}_j}} + \alpha_{i,j} u_j{}^{[n+1]} = -\left(\frac{\partial u_j{}^{[n+1]}}{\partial n_{\mathcal{L}_j}} + \alpha_{j,i} u_j{}^{[n+1]}\right) + (\alpha_{j,i} + \alpha_{i,j}) u_j{}^{[n+1]}$$

As $u_j^{[n+1]} = \mathcal{M}_j(\boldsymbol{\xi}_j^{[n]}, \boldsymbol{\mathcal{F}}_j)$ and from (2.17) on $\Gamma_{j,i}$ we obtain

$$\frac{\partial u_j{}^{[n+1]}}{\partial n_{\mathcal{L}_j}} + \alpha_{j,i} u_j{}^{[n+1]} = \xi_{j,i}^{[n]} := -\frac{\partial u_i{}^{[n]}}{\partial n_{\mathcal{L}_i}} + \alpha_{j,i} u_i{}^{[n]}$$

and then $\forall (i,j) \in [\![1, M]\!]^2$, such that $\Gamma_{i,j} \neq \varnothing$ we have

$$\xi_{i,j}^{[n+1]} = -\xi_{j,i}^{[n]} + (\alpha_{j,i} + \alpha_{i,j}) u_j{}^{[n+1]} \tag{5.2}$$

## 5.1 Matlab implementation

We now explain a step of the iteration process in Matlab using mooVecFEM toolbox ($P_1$ Lagrange finite elements). With Matlab, we can use array of cell data to store discretisation of $\boldsymbol{u} = (u_1, \dots, u_M) \in \mathrm{H}^1(\Omega_1) \times \dots \times \mathrm{H}^1(\Omega_M)$ and $\boldsymbol{\xi} = (\xi_{i,j})_{i,j=1}^{M} \in \mathrm{L}^2(\Gamma^{\mathrm{int}})$ :

- `U` is an array of $M$ cells. `U{i}` is a vector of dimension the number of vertices of $\Omega_i$.
- `Xi` is a $M-by-M$ matrix of cells. if $\Gamma_{i,j} \neq \varnothing$, `Xi{i,j}` is then a vector of dimension the number of vertices of $\Gamma_{i,j}$, otherwise `Xi{i,j}` is empty.
- `Alpha` is a $M-by-M$ matrix of cells. if $\Gamma_{i,j} \neq \varnothing$, `Alpha{i,j}` is either a scalar or a vector of dimension the number of vertices of $\Gamma_{i,j}$.

Let `U`, `Xi` and `Alpha` respectively be the $P_1$-Lagrange discretization of $\boldsymbol{u}^{[n]}$, $\boldsymbol{\xi}^{[n]}$ and $\boldsymbol{\alpha}$. Then to compute discretization of $\boldsymbol{u}^{[n+1]}$, $\boldsymbol{\xi}^{[n+1]}$ also store as cell arrays `UN` and `XiN`, we first have to set the Robin transmission conditions on all partitions:

$$\frac{\partial u_i^{[n+1]}}{\partial n_i} + \alpha_{i,j} u_i^{[n+1]} = \xi_{i,j}^{[n]}, \qquad \text{on } \Gamma_{i,j}, \quad \forall j \in [\![1, M]\!]. \tag{5.3}$$

This can be done using following command :

```
DDbvp.setDDRobin(Xi,Alpha);
```

Now we can compute a discretization of $\boldsymbol{u}^{[n+1]}$ :

```
UN=DDbvp.solve();
```

where solve is the object method function

```
function U=solve(self)
  U=cell(self.np,1);
  for i=1:self.np
    [A,b]=self.bvps{i}.Assembly('local',true,'interface',true);
    U{i}=A\b;
  end
end
```

To compute a discretization of $\boldsymbol{\xi}^{[n+1]}$ we do :

```
XiN=Bt(DDbvp,Xi,Alpha,idxTrace,UN);
```

where `idxTrace` is a $M-by-M$ cell matrix such that `UN{i}(idxTrace{i,j})` is the trace of `UN{i}` on $\Gamma_{i,j}$ and the function Bt is given, using (5.2), by

```
function LambdaN=Bt(DDbvp,Lambda,Alpha,idxTrace,U)
  for i=1:DDbvp.np
    for j=1:DDbvp.np
      if DDbvp.ThDD.InterElts{1}.Gamma(i,j)~=0
        LambdaN{i,j}=-Lambda{j,i}+ ...
            (Alpha{j,i}+Alpha{i,j}).*U{j}(idxTrace{j,i});
      end
```

```
    end
  end
end
```

Listing 5: function Bt

We now give the Matlab code (simple without optimization) to compute the 200 first iterates of the fixed point algorithm.

```
Xi=cell(DDbvp.np,DDbvp.np);
for n=1:200
  DDbvp.setDDRobin(Xi,Alpha);
  U=DDbvp.solve();
  Xi=Bt(DDbvp,Xi,Alpha,idxTrace,U);
end
```

One can also write the fixed point function Sbase given in Listing 8 and then use the fixed point algorithm (FPA) given in Listing 30 to compute a numerical approximation of (3.7) :

```
Phi=@(X) Sbase(bvp,X,Alpha,idxTrace);
XiN=initXi(bvp);
[Xi,info]=cellFixedPoint(Phi,XiN,'itermax',200,'tol',1e-8);
```

Listing 6: Used of fixed point algorithm to solve the fixed point interface problem (3.7)

```
function varargout=Sbase(DDbvp,Xi,Alpha,idxTrace)
  DDbvp.setDDRobin(Xi,Alpha);
  U=DDbvp.solve();
  XiN=Bt(DDbvp,Xi,Alpha,idxTrace,U);
  varargout{1}=XiN;
  if nargout==2,varargout{2}=U;end
end
```

Listing 7: function Sbase

```
function varargout=cellFixedPoint(Phi,x0,varargin)
  p = inputParser;
  p.addParamValue('tol', 1e-6, @isscalar );
  p.addParamValue('itermax', 200, @isscalar );
  p.addParamValue('verbose', false, @islogical );
  p.parse(varargin{:});
  R=p.Results;
  tol=R.tol;itermax=R.itermax;
  err=zeros(itermax,1);
  k=1;err(1)=tol+1;Xtol=[];
  if nargout==3, Xs=cell(itermax+1,1);Xs{1}=x0;end

  X=x0;Xp=x0;
  while ( (err(k)>tol) && (k<=itermax) )
    Xp=X;
    X=Phi(Xp);
    k=k+1;
    if nargout==3, Xs{k}=X;end
```

```
19      err(k)=ErrorCell(X,Xp)/(ErrorCell(X,[])+1);
20      if R.verbose, fprintf('cellFixedPoint:  iteration %d,...
            error=%.4e\n',k-1,err(k));end
21    end
22    if nargout>=1, varargout{1}=X;end
23    if nargout>=2
24      varargout{2}=struct('err',err(2:k),'iter',k-1, ...
            'converge',(err(k)<=tol));
25    end
26    if nargout==3
27      varargout{3}=Xs;
28    end
29  end
```

Listing 8: cellFixedPoint

Obviously, one can compute and represent the solution by

```
1  DDbvp.setDDRobin(Xi,Alpha);
2  U=DDbvp.solve();
3  DDbvp.ThDD.plotVal(U)
```

## 5.2 Applications

### 5.2.1 BVP sample 1, 2-by-1 partitions

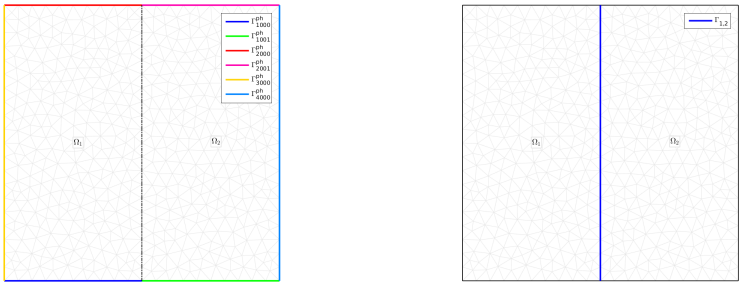


Figure 3: 2-by-1 partionned mesh (nq = 557), boundaries (left) and interface (right)

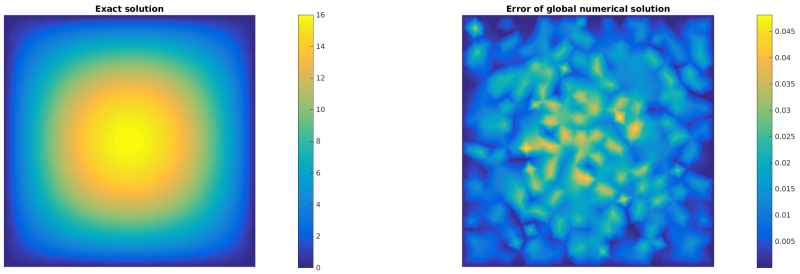



Figure 4: BVP sample 1 (nq = 557), exact solution (left) and global numerical solution (right)

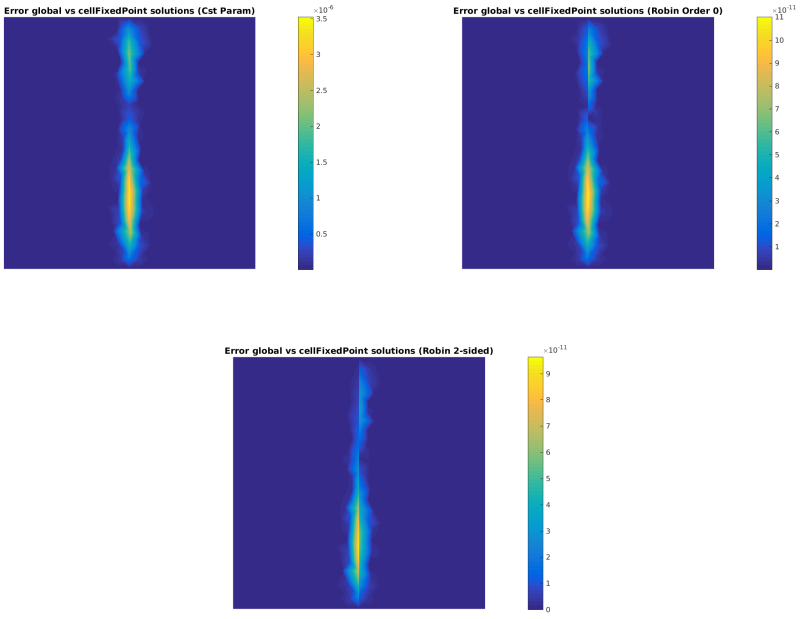



Figure 5: BVP sample 1 (nq = 557), error between global and FPA numerical solution with Robin constant parameters $\alpha = 1$ (upper left), optimal Robin order 0 (upper right) and 2-sided optimal Robin (bottom)

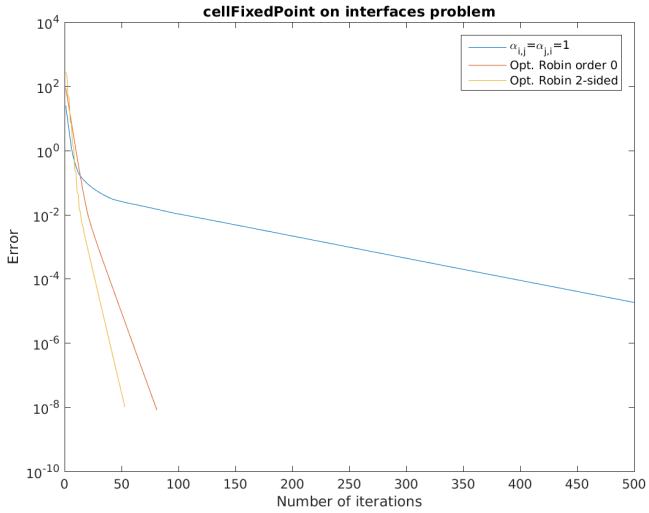



Figure 6: BVP sample 1 (nq = 557), error

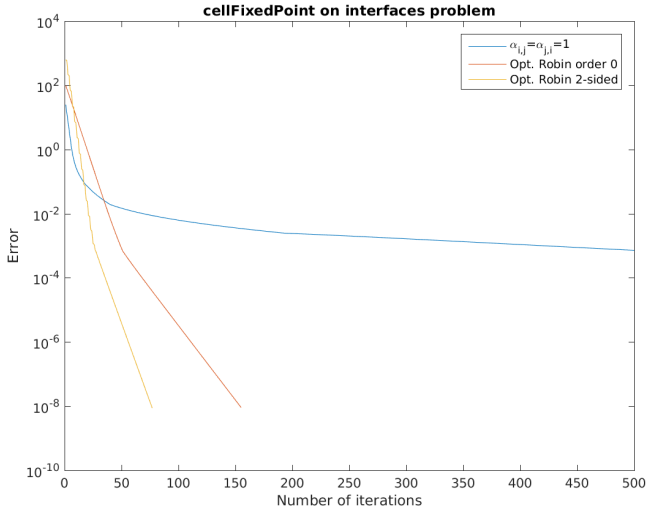



Figure 7: BVP sample 1, error with nq = 8653 ($N = 80$)

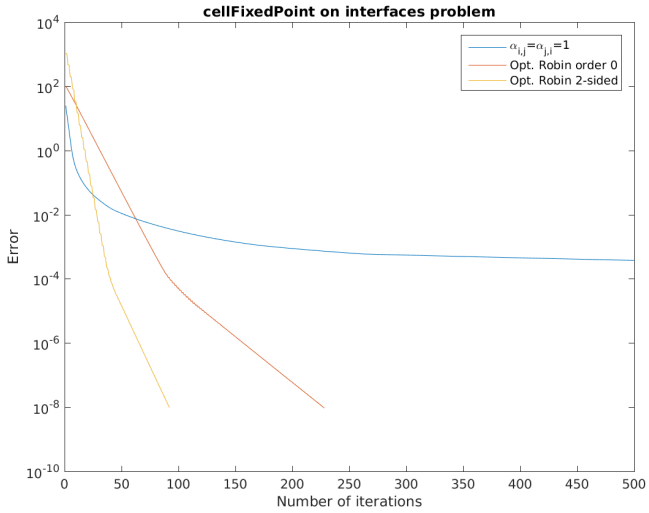



Figure 8: BVP sample 1, error with nq = 53400 ($N = 200$)

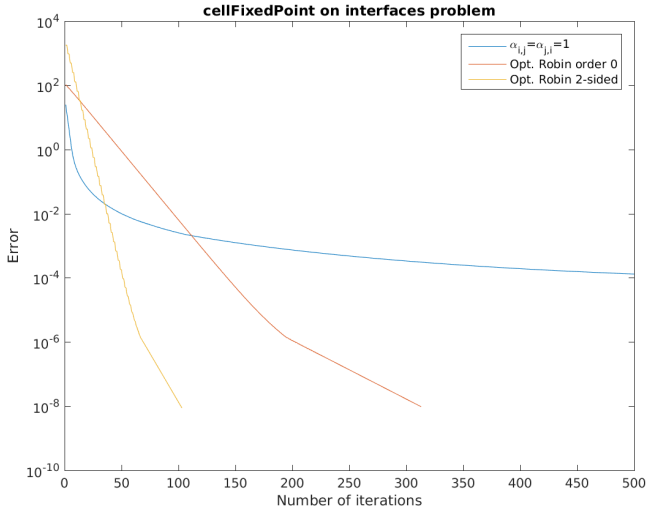



Figure 9: BVP sample 1, error with nq = 331610 ($N = 500$)

We show now how well the analysis predicts the optimization parameters in the nonoverlapping case.

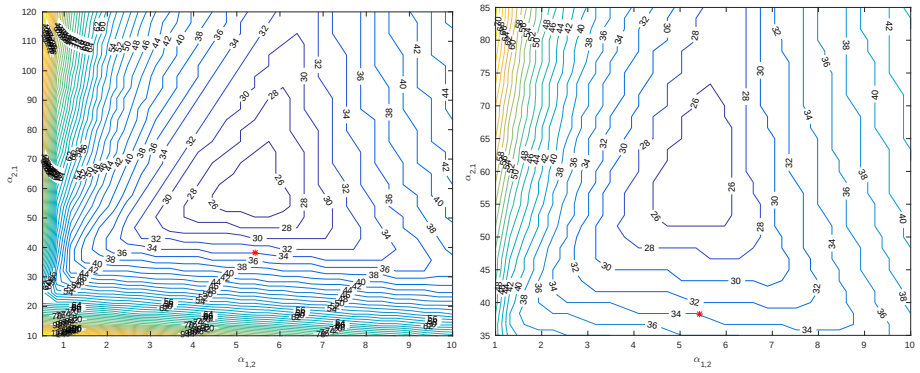


Figure 10: 2-by-1 partionned mesh (nq = 557), $\alpha_{1,2} \neq \alpha_{2,1}$. The red star is the predicted 2-sided optimal Robin parameters.

From [1], Corollary 4.10, formula (4.46), the asymptotic performance of the fixed point algorithm with 2-sided optimal Robin parameters is given by

$$\rho(h) = 1 - Ch^{\frac{1}{4}} + \mathcal{O}(h^{\frac{1}{2}})$$

with $C = 2\frac{\sqrt{2}(k_{\min}^2+\eta)^{\frac{1}{8}}}{\pi^{\frac{1}{4}}}$. So the error at iteration $k$, denoted by $e^{[k]}$ is given by

$$e^{[k]} \approx \rho(h)^k e^{[0]}.$$

The number of iterations needed to obtain the precison tol is the number $k$ such that

$$\rho(h)^k = \frac{\text{tol}}{e^{[0]}}.$$

As $\log(1-x) = -x - \frac{x^2}{2} + \mathcal{O}(x^3)$ for $x$ near 0, we obtain

$$\log(\rho(h)^k) = k\log(1 - Ch^{\frac{1}{4}}) \approx kCh^{\frac{1}{4}} \approx \log(\frac{\text{tol}}{e^{[0]}})$$

and then the number of iterations increase as $\mathcal{O}(h^{-\frac{1}{4}})$.

From [1], Theorem 4.5, formula (4.20), the asymptotic performance of the fixed point algorithm with optimized Robin transmission conditions (order 0) is given by

$$\rho(h) = 1 - Ch^{\frac{1}{2}} + \mathcal{O}(h)$$

with $C = 4\frac{(k_{\min}^2+\eta)^{\frac{1}{4}}}{\pi^{\frac{1}{2}}}$. So the number of iterations increase as $\mathcal{O}(h^{-\frac{1}{2}})$.

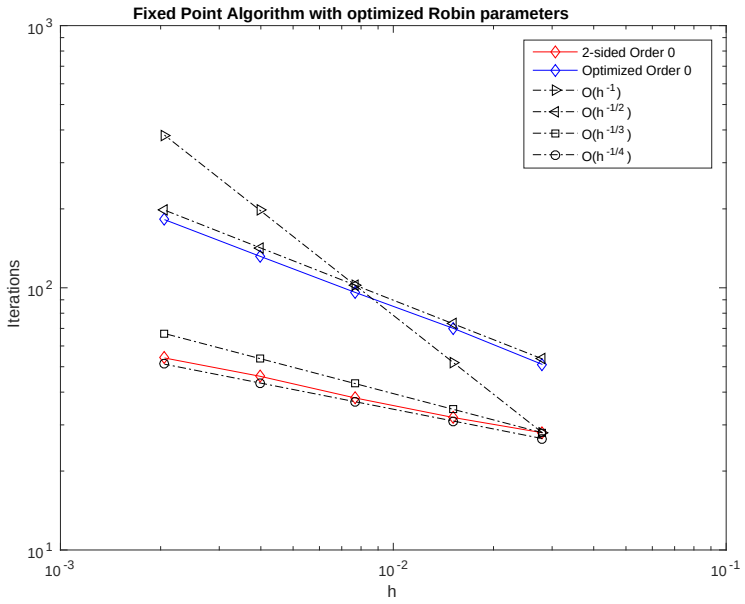



Figure 11: BVP sample 1, asymptotic number of iterations required by fixed point algorithms

We give in Appendix B.1 other numerical results for the unit square with 2-by-3, 8-by-1, 1-by-8, and 8-by-8 partitions.

### 5.2.2 BVP sample 1, 2-by-1 partitions : bug

We now choose exact solution none 0 on boundaries : $u_{\text{ex}}(x, y) = \cos(2x^2 - 6y^2)$

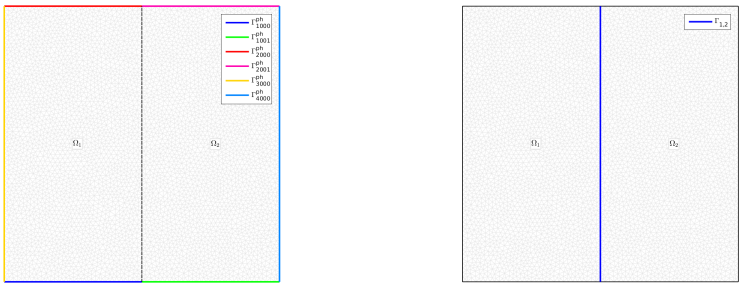


Figure 12: 2-by-1 partionned mesh (nq = 4823), boundaries (left) and interface (right)

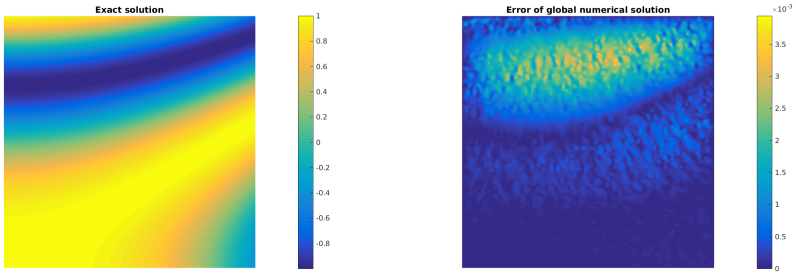



Figure 13: BVP sample 1 (nq = 4823), exact solution (left) and global numerical solution (right)

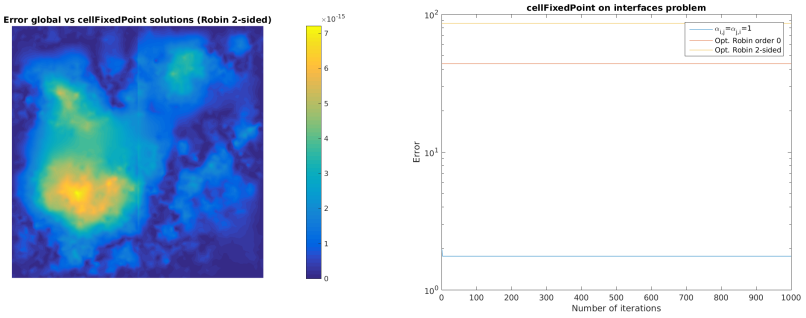



Figure 14: BVP sample 1 (nq = 4823), error between global and FPA numerical solution and 2-sided optimal Robin (left) and error of interfaces problem

From Figure 16, right picture, one can see that the algorithm of the fixed point on interfaces problem does not converge: the maximum number of iterations `maxiter`=1000 is reached. We have

$$Error(n+1) = \max_{i,j} \left\| \xi_{i,j}^{[n+1]} - \xi_{i,j}^{[n]} \right\|_{\infty} \approx 10^2.$$

However, from the left picture, we represented the error between the global solution (computed without DD) and the DD solution on $\Omega$ computed with $\boldsymbol{\xi}^{[1000]}$ : the DD solution on $\Omega$ converge to the global solution!

To understand this phenomenon numerically, we represent the last four iterates of $\boldsymbol{\xi}^{[n]}$ in Figure 15.

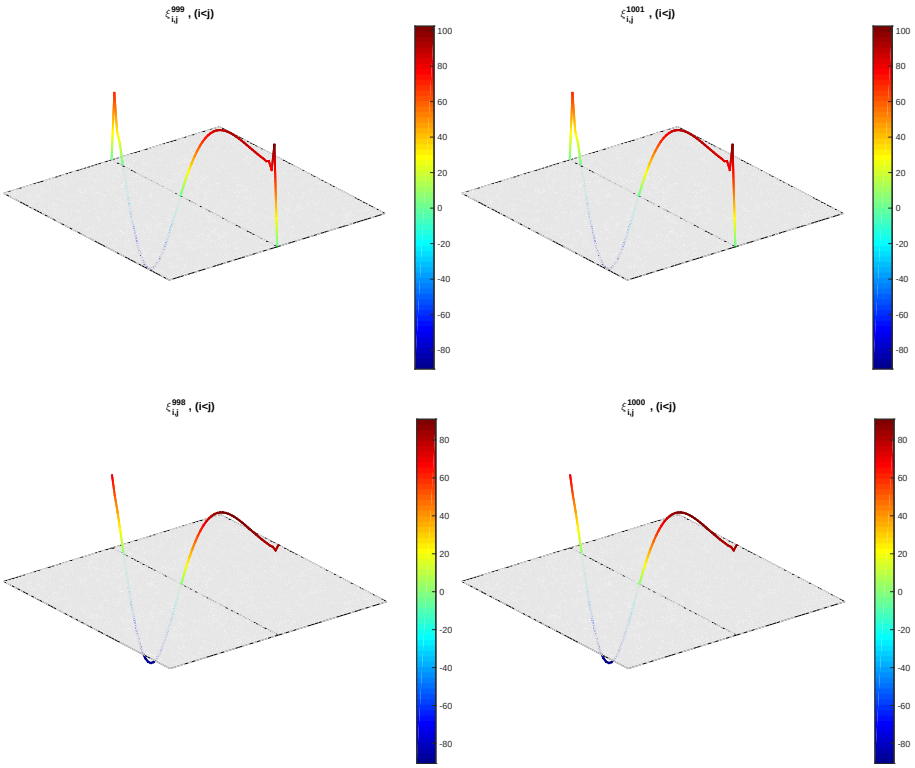


Figure 15: Interface solutions at iterations 998 to 1001

From Figure 15, it seems that the even and the odds iterates of $\boldsymbol{\xi}^{[n]}$ converges towards two distinct *values*. To obtain a convergent fixed point algorithm we write a new fixed point problem :

$$\boldsymbol{\xi}^{[n+1]} = \mathcal{S}(\mathcal{S}(\boldsymbol{\xi}^{[n]}, \boldsymbol{\mathcal{F}}), \boldsymbol{\mathcal{F}}), \ \forall n \geqslant 0 \tag{5.4}$$

with a given $\boldsymbol{\xi}^{[0]} \in \mathrm{L}^2(\Gamma^{\mathrm{int}})$.

The corresponding Matlab code :

```
Phi=@(X)  Sbase(bvp,X,Alpha,idxTrace);
Phi2=@(X)  Phi(Phi(X));
XiN=initXi(bvp);
[Xi,info]=cellFixedPoint(Phi2,XiN,'itermax',200,'tol',1e-8);
```

Listing 9: Used of fixed point algorithm to solve the new fixed point interface problem (??)

With this code we now obtain

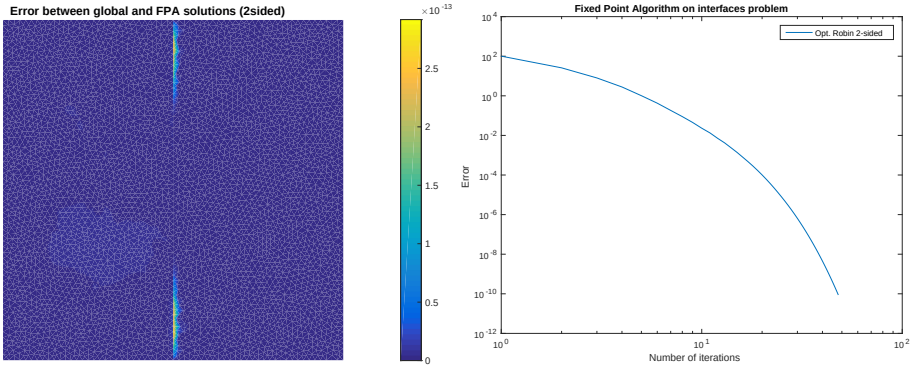



Figure 16: BVP sample 1 (nq = 4823), error between global and FPA numerical solution for problem (5.4))with 2-sided optimal Robin (left). Error of interfaces problem (right).

An other way, is to compute the iteration error as $\left\|\boldsymbol{u}^{[n+1]} - \boldsymbol{u}^{[n]}\right\|_\infty$ instead of $\left\|\boldsymbol{\xi}^{[n+1]} - \boldsymbol{\xi}^{[n]}\right\|_\infty$.

### 5.2.3 BVP sample 2, 2 partitions

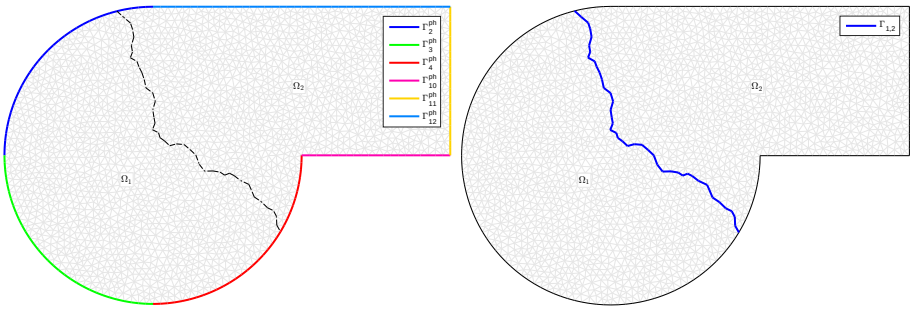


Figure 17: Mesh with $n_p = 2$ partitions (nq = 9408), boundaries (left) and interface (right)

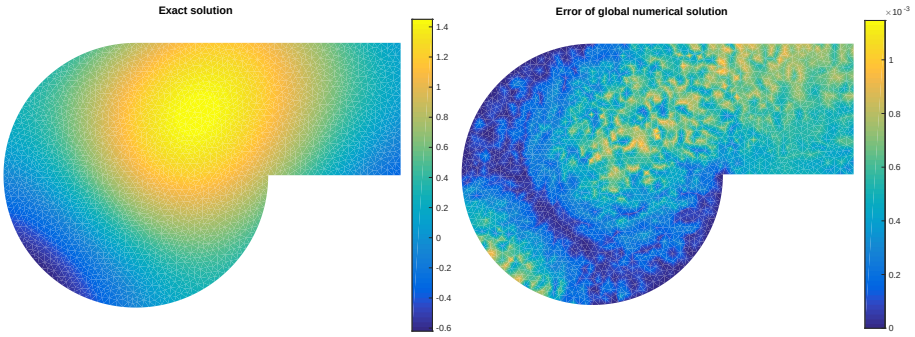



Figure 18: BVP sample 2 ($n_p = 2$, nq = 9408), exact solution (left) and global numerical solution (right)

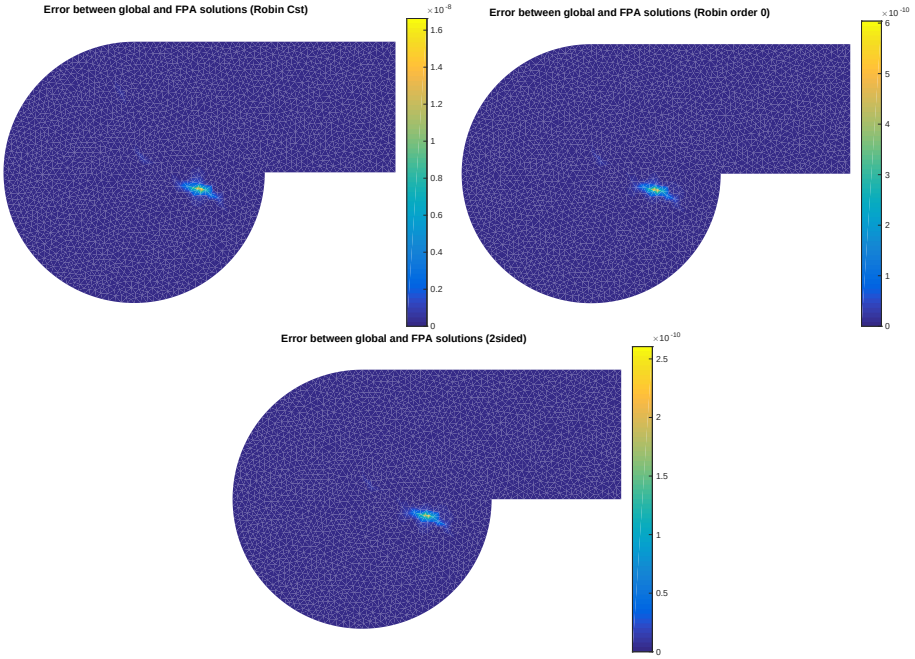



Figure 19: BVP sample 2 (nq = 9408), error between global and FPA numerical solution with Robin constant parameters $\alpha = 1$ (upper left), optimal Robin order 0 (upper right) and 2-sided optimal Robin (bottom)

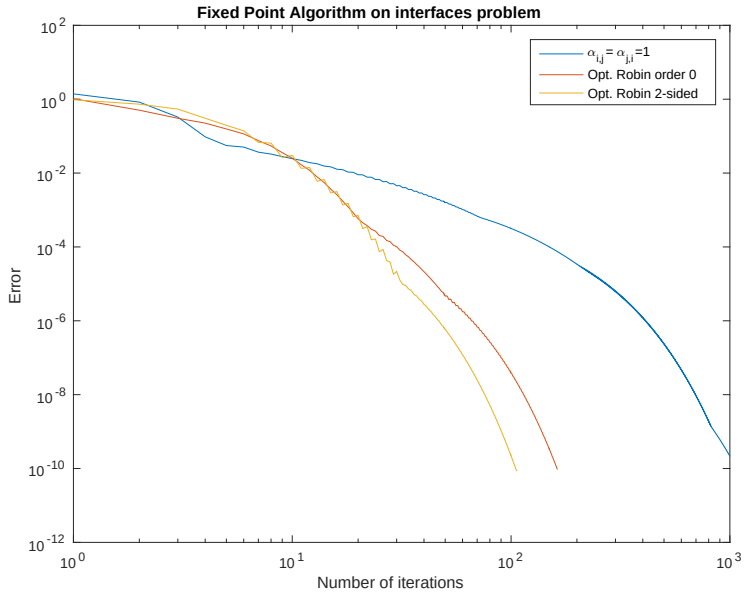



Figure 20: BVP sample 1 (nq = 9408), error

We show now how well the analysis predicts the optimization parameters in the nonoverlapping case.

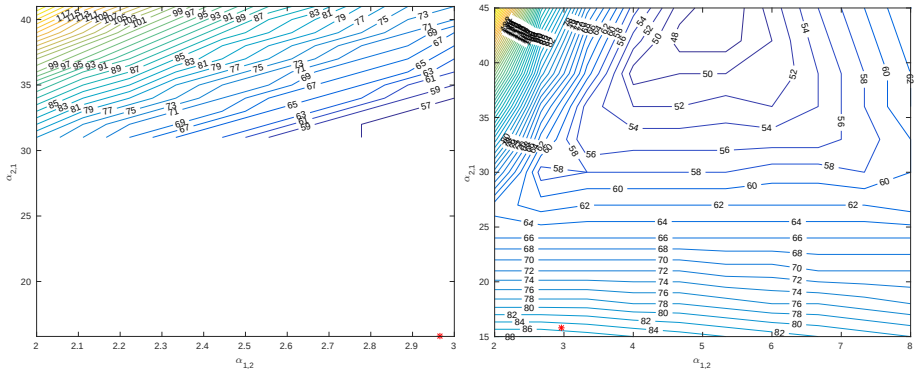


Figure 21: 2-by-1 partionned mesh (nq = 2432), $\alpha_{1,2} \neq \alpha 2, 1$. The red star is the predicted 2-sided optimal Robin parameters.

# 6 Krylov methods

Due to linearity of $\mathcal{M}$ and $\mathcal{B}$ operators the fixed point interface problem (3.7) can be rewritten as

$$\begin{aligned} \boldsymbol{\xi} &= \mathcal{S}(\boldsymbol{\xi}, \boldsymbol{\mathcal{F}}) = (\mathcal{B} \circ \mathcal{M}(\boldsymbol{\xi}, \boldsymbol{\mathcal{F}}))^{\mathtt{t}} = (\mathcal{B} \circ \mathcal{M}(\boldsymbol{\xi}, 0))^{\mathtt{t}} + (\mathcal{B} \circ \mathcal{M}(0, \boldsymbol{\mathcal{F}}))^{\mathtt{t}} \\ &= \mathcal{S}(\boldsymbol{\xi}, 0) + \mathcal{S}(0, \boldsymbol{\mathcal{F}}) \end{aligned}$$

and thus we have to find $\boldsymbol{\xi}$ solution of

$$\mathcal{A}(\boldsymbol{\xi}) = \boldsymbol{b} \tag{6.1}$$

where $\mathcal{A}(\boldsymbol{\xi}) \stackrel{\text{def}}{=} \boldsymbol{\xi} - \mathcal{S}(\boldsymbol{\xi}, 0)$ and $\boldsymbol{b} \stackrel{\text{def}}{=} \mathcal{S}(0, \boldsymbol{\mathcal{F}})$.

One can solve this equation using Krylov methods.

## 6.1 Matlab implementation

We first give the syntax of some Matlab functions based on iterative methods to solve (6.1)

BICGSTAB : **BiConjugate Gradients Stabilized Method.**

- x = bicgstab(A,b); attempts to solve the system of linear equations A*x=b for x. The n-by-n coefficient matrix A must be square and should be large and sparse. The column vector b must have length n. A can be a function handle, afun, such that afun(x) returns A*x.

BICGSTABL : **BiConjugate Gradients Stabilized(l) Method.**

- X = bicgstabl(A,B); attempts to solve the system of linear equations A*X=B for X. The N-by-N coefficient matrix A must be square and the right hand side column vector B must have length N.
- X = bicgstabl(AFUN,B); accepts a function handle AFUN instead of the matrix A. AFUN(X) accepts a vector input X and returns the matrix-vector product A*X. In all of the following syntaxes, you can replace A by AFUN.

CGS : **Conjugate Gradients Squared Method.**

GMRES : **Generalized Minimum Residual Method.**

LSQR : **LSQR Method.**

MINRES : **Minimum Residual Method.**

PCG : **Preconditioned Conjugate Gradients Method.**

QMR : **Quasi-Minimal Residual Method.**

To use one of theses functions, we must transform the unknow data `Xi`, which was implemented as a cell matrix (see section 5.1), into a vector `XiVec`. That's done by function InterCell2Vec given in Listing 20. The inverse function InterVec2Cell is given in Listing 16.

```
1  function interVec=InterCell2Vec(DDbvp,interCell)
2    ThDD=DDbvp.ThDD;
3    interVec=zeros(DDbvp.ThDD.InterElts{1}.nq,1);
4    start=1;
5    for i=1:DDbvp.np
6      for j=1:DDbvp.np
7        idx=DDbvp.ThDD.InterElts{1}.GammaIdx(i,j);
8        if idx~=0
```

```
          nq=DDbvp.ThDD.Th{i}.sTh{idx}.nq;
          if length(interCell{i,j})==1
            interVec(start:start+nq-1)=ones(nq,1)*interCell{i,j};
          else
            interVec(start:start+nq-1)=interCell{i,j};
          end
          start=start+nq;
        end
      end
    end
end
```

Listing 10: function InterCell2Vec

```
function interCell=InterVec2Cell(DDbvp,interVec)
  ThDD=DDbvp.ThDD;
  interCell=cell(DDbvp.np,DDbvp.np);
  start=1;
  for i=1:DDbvp.np
    for j=1:DDbvp.np
      idx=DDbvp.ThDD.InterElts{1}.GammaIdx(i,j);
      if idx~=0
        nq=DDbvp.ThDD.Th{i}.sTh{idx}.nq;
        interCell{i,j}=interVec(start:start+nq-1);
        start=start+nq;
      end
    end
  end
end
```

Listing 11: function InterVec2Cell

To solve problem (6.1), we must differentiate computation of $\mathcal{S}(0,\boldsymbol{\mathcal{F}})$ and $\mathcal{S}(\boldsymbol{\xi},0)$. The first one is immediately obtain with

```
Xi=initXi(bvp);  % set Xi to "0"
bcell=Sbase(bvp,Xi,Alpha,idxTrace); % cell data
b=InterCell2Vec(bvp,bcell);
```

The second one is a little more complicate to compute : we must write a new Sbase function where all the source terme in the $M$ BVP, except Robin source on the interfaces, are taken to zero (i.e. $\boldsymbol{\mathcal{F}} = 0$). For that on each local BVP we compute $u_i = \mathcal{M}(\boldsymbol{\xi},0)$ using the following algorithm :

1: $[AI, bI] \leftarrow$ Assemby of interfaces contributions of $\Omega_i$.
2: $[A, bF] \leftarrow$ Assemby of all contributions on $\Omega_i$ except interfaces and Dirichlet.
3: $[ID, IDc, gD] \leftarrow$ Assemby of Dirichlet contributions.
4: $gD \leftarrow 0$,
5: $A \leftarrow A + AI$, $b \leftarrow bI$
6: $U_i \leftarrow$ Solve linear system $A * x = b$ with Dirichlet conditions given with (ID,IDc,gD)

That's give the Matlab function Mbase0 given in Listing 14

```
function U=Mbase0(bvps)
```

```
  np=length(bvps);
  U=cell(np,1);
  for i=1:np
    [AI,bI]=bvps{i}.Assembly('local',true,'dom',false, ...
        'Dirichlet',false,'physical',false,'interface',true);
    [A,~]=bvps{i}.Assembly('local',true,'Dirichlet',false);
    [ID,IDc,gD]=bvps{i}.AssemblyDirichlet();
    A=A+AI;
    b=bI;
    gD=0*gD;
    U{i}=classicSolve(A,b,bvps{i}.ndof,gD,ID,IDc);
  end
end
```

Listing 12: function MBASE0

So the function SVECBASE0 which compute $\mathcal{S}(\boldsymbol{\xi}, 0)$ is now easy to write and its given in Listing 14.

```
function varargout=SvecBase0(DDbvp,XiVec,Alpha,idxTrace)
  % with sources terme to 0
  global iter
  iter=iter+1;
  fprintf('  iter=%d\n',iter)
  Xi=InterVec2Cell(DDbvp,XiVec);
  DDbvp.setDDRobin(Xi,Alpha);
  [XiN,U]=Sbase0(DDbvp,Xi,Alpha,idxTrace);
  XiVecN=InterCell2Vec(DDbvp,XiN);
  varargout{1}=XiVecN;
  if nargout==2,varargout{2}=U;end
end
```

Listing 13: function SVECBASE0

With theses functions, one can used Matlab BICGSTAB to solve problem (6.1):

```
Xi=initXi(bvp);  % set Xi to "0" (cell)
bcell=Sbase(bvp,Xi,Alpha,idxTrace); % cell data
b=InterCell2Vec(bvp,bcell);
A=@(x) x-SvecBase0(bvp,x,Alpha,idxTrace);
[XiVec,flag,relres,nbiter]=bicgstab(A,b,tol,itermax);
```

## 6.2 Applications

### 6.2.1 BVP sample 1, 2-by-1 partitions

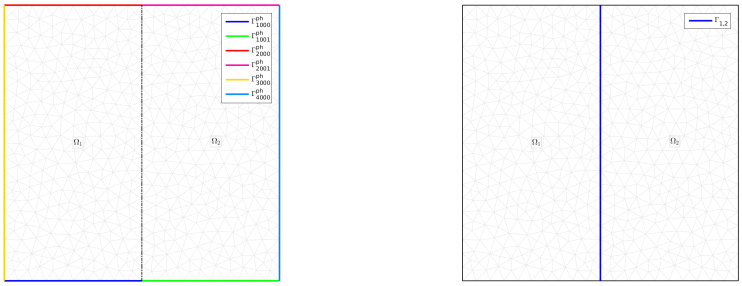


Figure 22: 2-by-1 partionned mesh (nq = 557), boundaries (left) and interface (right)

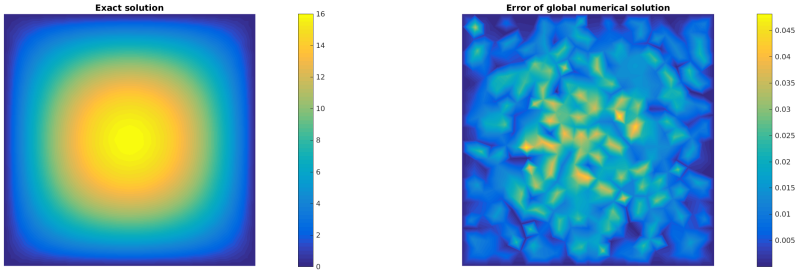



Figure 23: BVP sample 1 (nq = 557), exact solution (left) and global numerical solution (right)

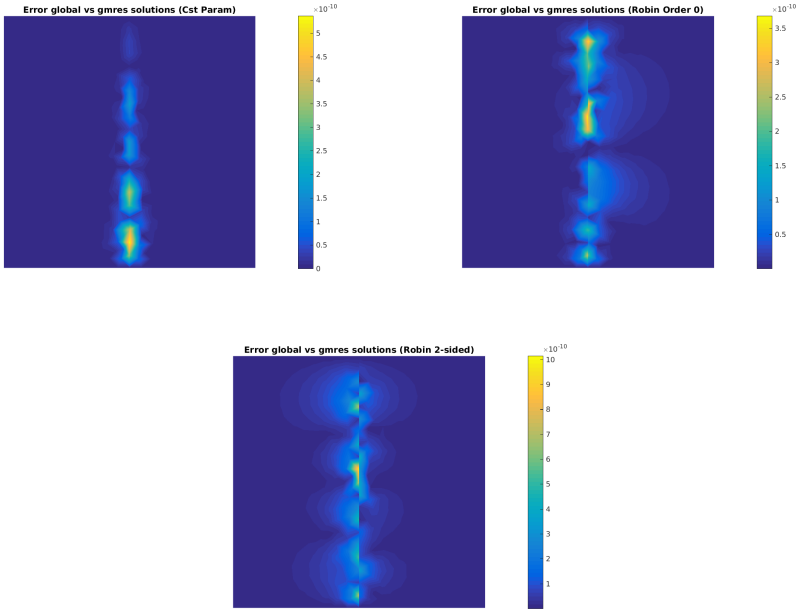



Figure 24: BVP sample 1 (nq = 557), error between global and gmres numerical solution with Robin constant parameters $\alpha = 1$ (upper left), optimal Robin order 0 (upper right) and 2-sided optimal Robin (bottom)

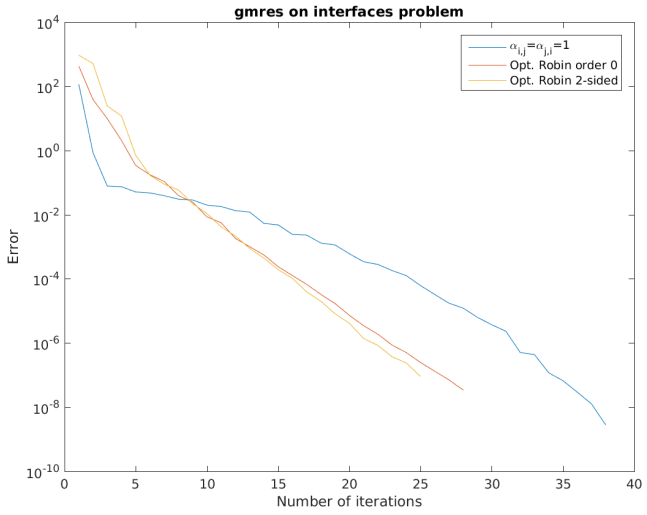



Figure 25: BVP sample 1 (nq = 557), error

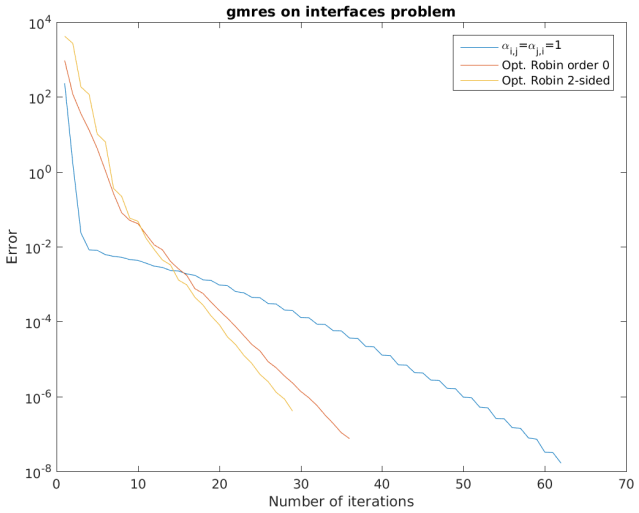



Figure 26: BVP sample 1, error with nq = 8653 ($N = 80$)

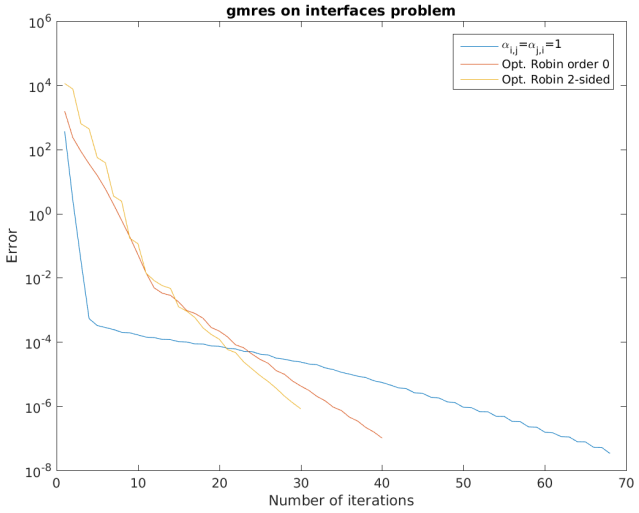



Figure 27: BVP sample 1, error with nq = 53400 ($N = 200$)

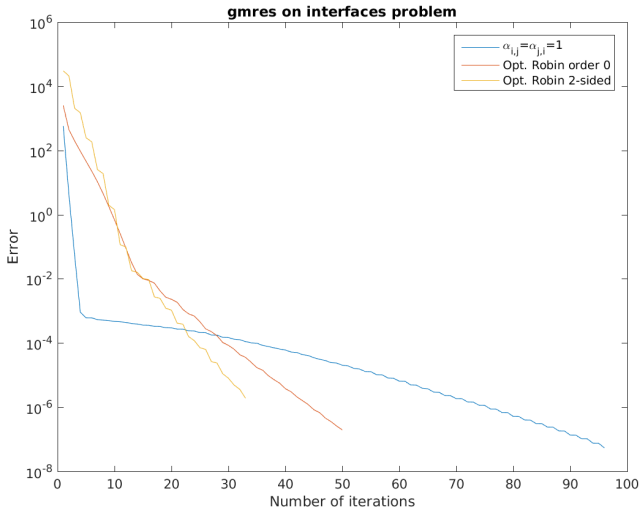



Figure 28: BVP sample 1, error with nq = 331610 ($N = 500$)

### 6.2.2 BVP sample 1, 2-by-3 partitions

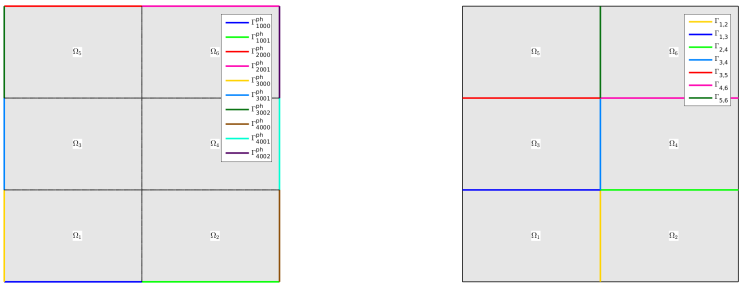


Figure 29: 2-by-3 partionned mesh (nq = 331610), boundaries (left) and interface (right)

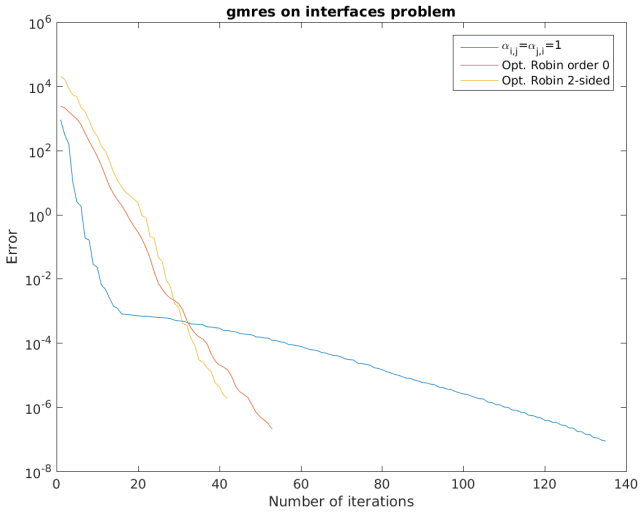



Figure 30: BVP sample 1, error with nq = 331610 ($N = 500$)

### 6.2.3 BVP sample 2, 2 partitions

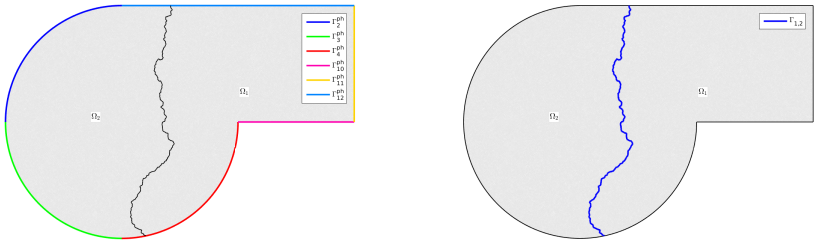


Figure 31: Mesh with $n_p = 2$ partitions (nq = 58399), boundaries (left) and interface (right)

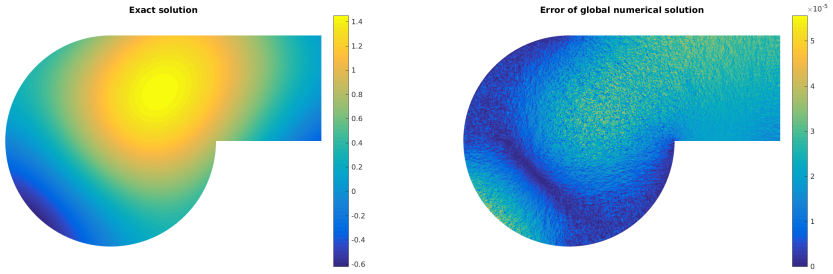



Figure 32: BVP sample 2 ($n_p = 2$, nq = 58399), exact solution (left) and global numerical solution (right)

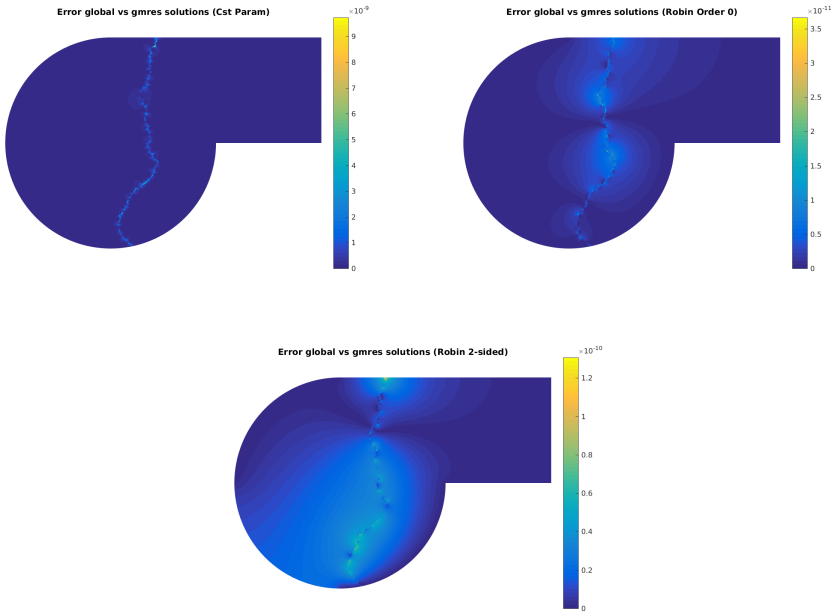



Figure 33: BVP sample 2 ($n_p = 2$, nq = 58399), error between global and GMRES numerical solution with Robin constant parameters $\alpha = 1$ (upper left), optimal Robin order 0 (upper right) and 2-sided optimal Robin (bottom)

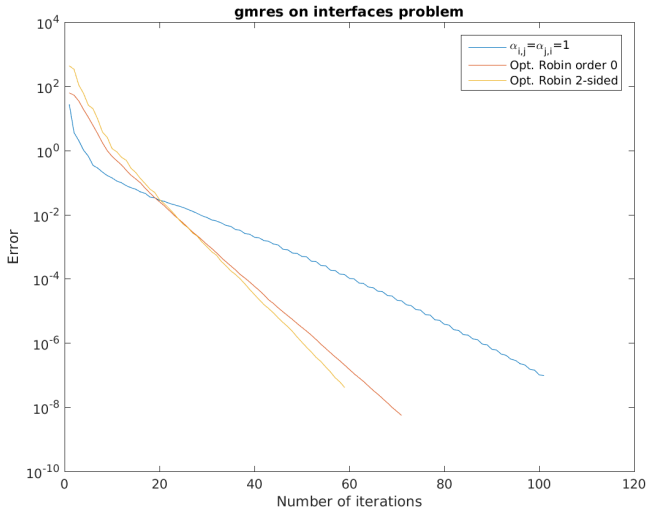



Figure 34: BVP sample 1 ($n_p = 2$, nq = 58399), error

### 6.2.4 BVP sample 2, 7 partitions

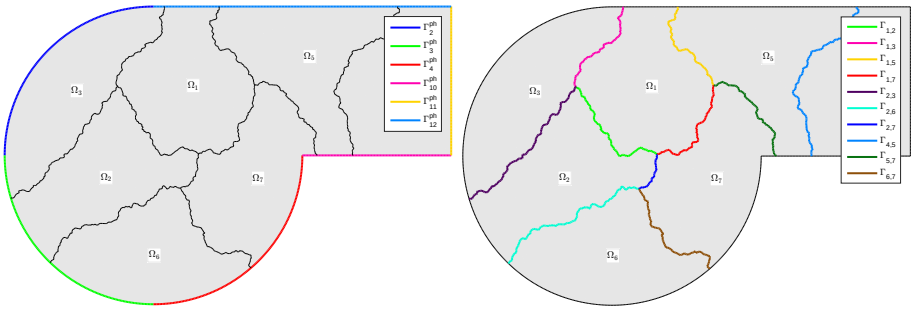


Figure 35: Mesh with $n_p = 7$ partitions (nq = 58399), boundaries (left) and interface (right)

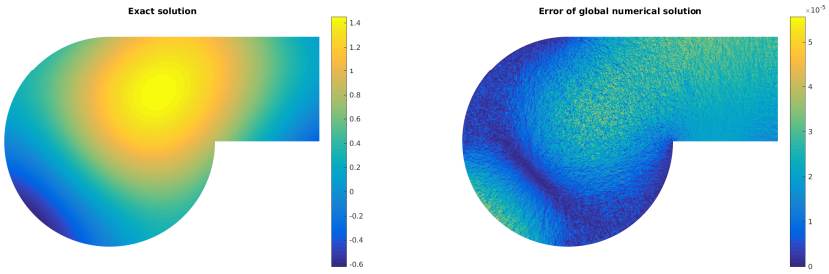



Figure 36: BVP sample 2 ($n_p = 7$, nq = 58399), exact solution (left) and global numerical solution (right)

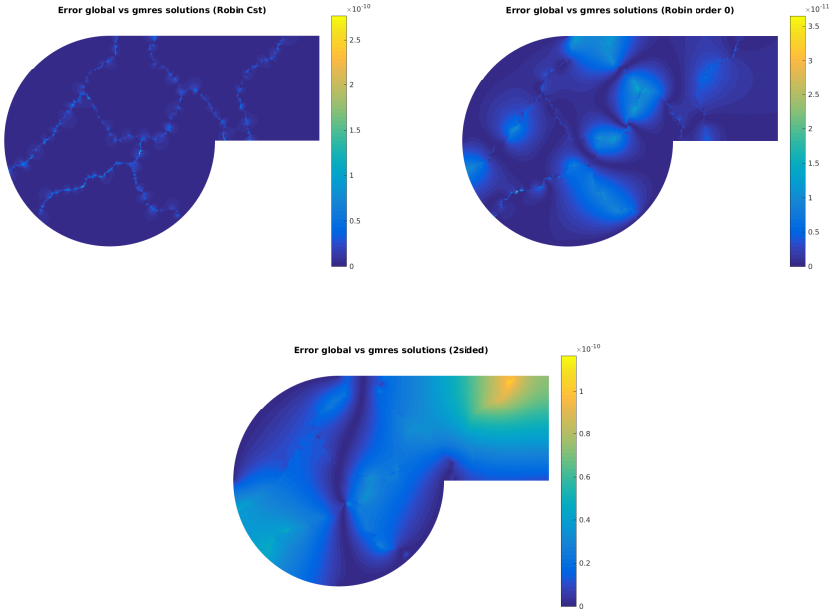



Figure 37: BVP sample 2 ($n_p = 7$, nq = 58399), error between global and GMRES numerical solution with Robin constant parameters $\alpha = 1$ (upper left), optimal Robin order 0 (upper right) and 2-sided optimal Robin (bottom)

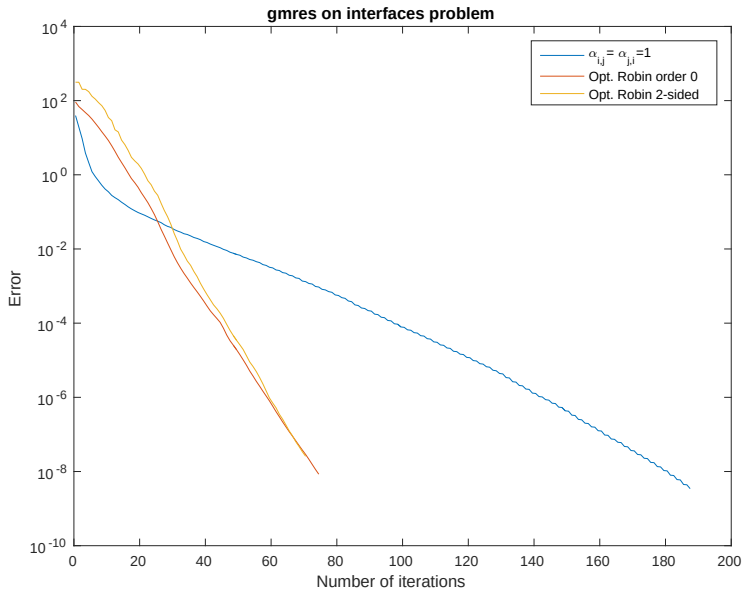



Figure 38: BVP sample 1 ($n_p = 7$, nq = 58399), error

### 6.2.5 BVP sample 2, 25 partitions

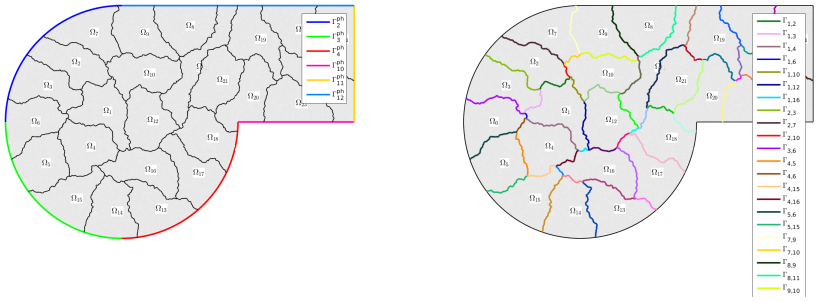


Figure 39: Mesh with $n_p = 25$ partitions (nq = 58399), boundaries (left) and interface (right)

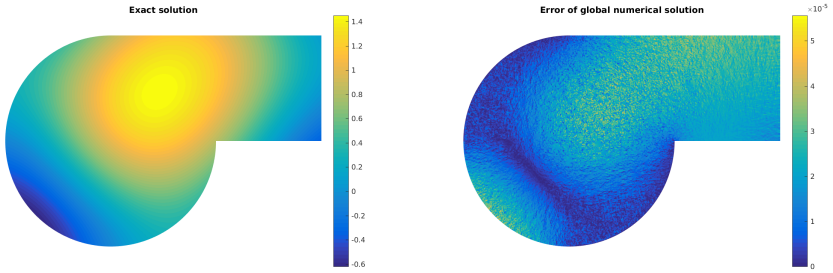



Figure 40: BVP sample 2 ($n_p = 25$, nq = 58399), exact solution (left) and global numerical solution (right)

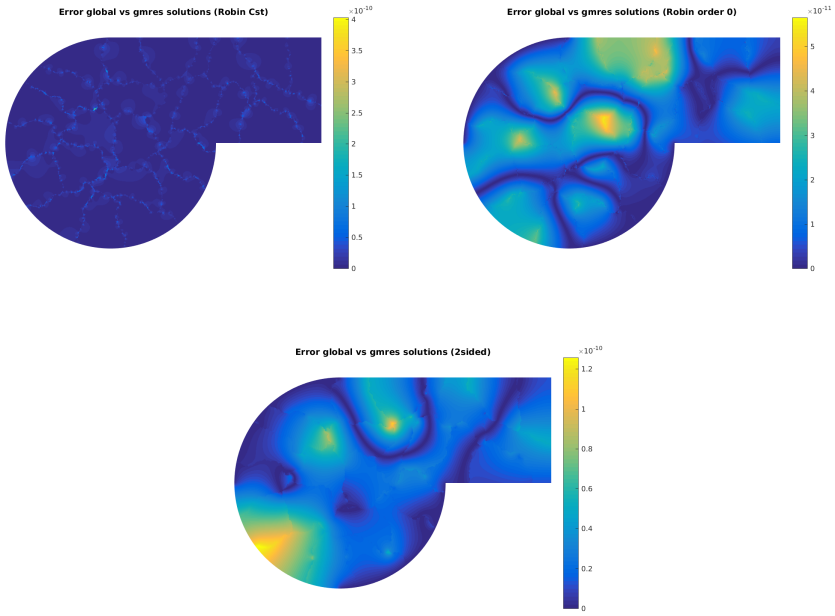



Figure 41: BVP sample 2 ($n_p = 25$, nq = 58399), error between global and GMRES numerical solution with Robin constant parameters $\alpha = 1$ (upper left), optimal Robin order 0 (upper right) and 2-sided optimal Robin (bottom)

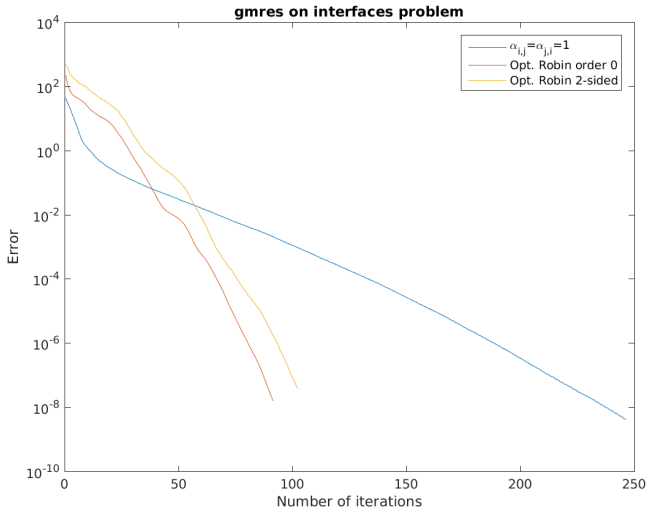



Figure 42: BVP sample 1 ($n_p = 25$, nq = 58399), error

# A Listings

## A.1 Computing Robin parameters

```
function Alpha=RobinParamCst(DDbvp,alpha)
  Alpha=cell(DDbvp.np,DDbvp.np);
  ThDD=DDbvp.ThDD;
  for i=1:DDbvp.np
    for j=1:DDbvp.np
      if ThDD.InterElts{1}.Gamma(i,j)~=0
        Alpha{i,j}= alpha;
      end
    end
  end
end
```

Listing 14: function RobinParamCst

```
function Alpha=OptRobinParamOrder0(DDbvp,eta)
  Alpha=cell(DDbvp.np,DDbvp.np);
  ThDD=DDbvp.ThDD;
  for i=1:DDbvp.np
    for j=i+1:DDbvp.np
      if ThDD.InterElts{1}.Gamma(i,j)~=0
        idxlab=ThDD.Th{i}.find(ThDD.Th{i}.d-1,ThDD.InterElts{1}.Gamma(i,j));
        h=ThDD.Th{i}.sTh{idxlab}.h;
        L=sum(ThDD.Th{i}.sTh{idxlab}.vols);
        kmin=pi/L;
        kmax=(pi)/h;
```

```
12            ps=OptimalRobinOrder0(eta,kmin,kmax);
13            Alpha{i,j}=ps;
14            Alpha{j,i}=ps;
15          end
16        end
17      end
18    end
19
20    % Compute p^*  Gander2006, formula 4.15
21    function [ps]=OptimalRobinOrder0(eta,kmin,kmax)
22      ps=((kmin^2+eta).*(kmax^2+eta)).^(1/4);
23    end
```

Listing 15: function OptRobinParamOrder0

```
1     function Alpha=OptRobinParamTwoSided(DDbvp,eta)
2       Alpha=cell(DDbvp.np,DDbvp.np);
3       %etah=1;
4       %etah=0;
5       ThDD=DDbvp.ThDD;
6       for i=1:DDbvp.np
7         for j=i+1:DDbvp.np
8           if ThDD.InterElts{1}.Gamma(i,j)~=0
9             idxlab=ThDD.Th{i}.find(ThDD.Th{i}.d-1,ThDD.InterElts{1}.Gamma(i,j));
10            h=ThDD.Th{i}.sTh{idxlab}.h;
11            %L=sum(ThDD.Th{i}.sTh{idxlab}.vols);
12            %B=ThDD.Th{i}.sTh{idxlab}.bbox;
13    %           B=ThDD.Th{i}.bbox;
14    %           h=ThDD.get_h();
15            L=norm(B(1:2:end)-B(2:2:end),2);
16            kmin=pi/L;
17            %kmax=(pi)/(2*h);% pi/(2*h) sembme pas mal
18            kmax=pi/h;
19            [ps,qs]=OptimalSecondOrder(eta,kmin,kmax);
20            [Alpha{i,j},Alpha{j,i}]=TwoSidedOptimizedRobin(eta,ps,qs);
21
22          end
23        end
24      end
25    end
26
27    % Compute p^* and q^* Gander2006, formula 4.30
28    function [ps,qs]=OptimalSecondOrder(eta,kmin,kmax)
29      a=kmax^2;b=kmin^2;A=sqrt(a+eta);B=sqrt(b+eta);
30      ps=(a*B-b*A)/(sqrt(2*(a-b))*((A-B)*((a+eta)*B-(b+eta)*A))^(1/4));
31      qs=(A-B)^(3/4)/(sqrt(2*(a-b))*((a+eta)*B-(b+eta)*A)^(1/4));
32    end
33
34    % Compute p_1^* and p_2^* Gander2006, formula 4.42
35    function [p1s,p2s]=TwoSidedOptimizedRobin(eta,ps,qs)
36      S=sqrt(1+4*eta*qs^2-4*ps*qs);
37      p1s=(1-S)/(2*qs);
38      p2s=(1+S)/(2*qs);
39    end
```

Listing 16: function OptRobinParamTwoSided

# B Numerical samples

## B.1 BVP sample 1, more partitioned mesh

### B.1.1 BVP sample 1, 2-by-3 partitions

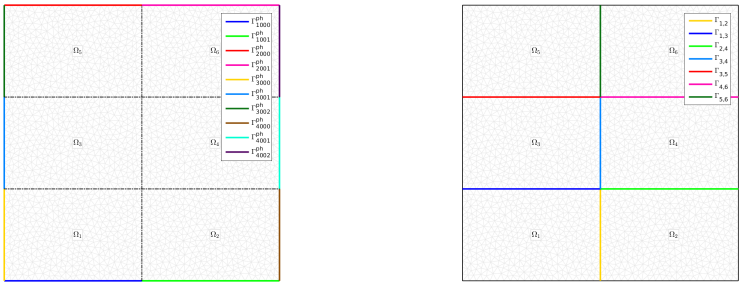


Figure 43: 2-by-3 partionned mesh (nq = 2309), boundaries (left) and interface (right)

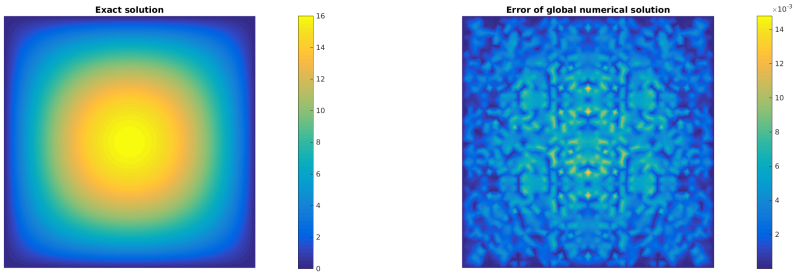



Figure 44: BVP sample 1 (nq = 2309), exact solution (left) and global numerical solution (right)

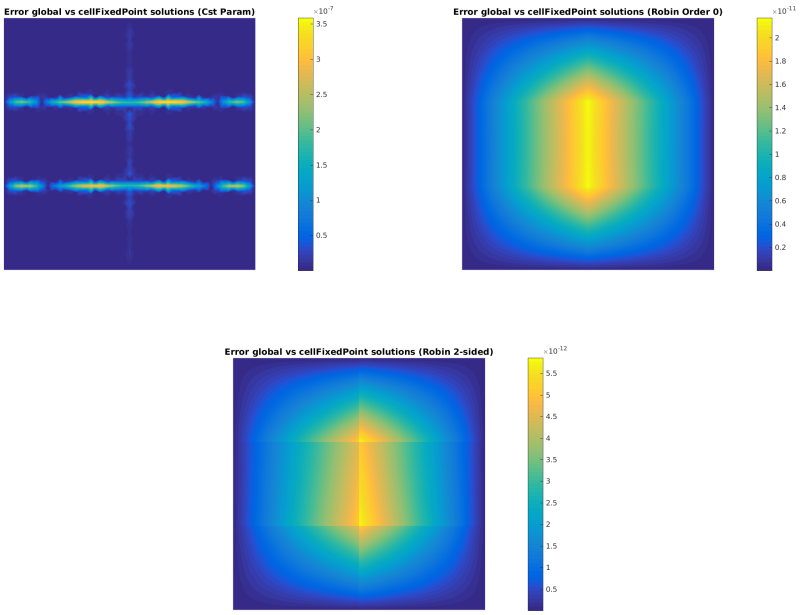



Figure 45: BVP sample 1 (nq = 2309), error between global and FPA numerical solution with Robin constant parameters $\alpha = 1$ (upper left), optimal Robin order 0 (upper right) and 2-sided optimal Robin (bottom)

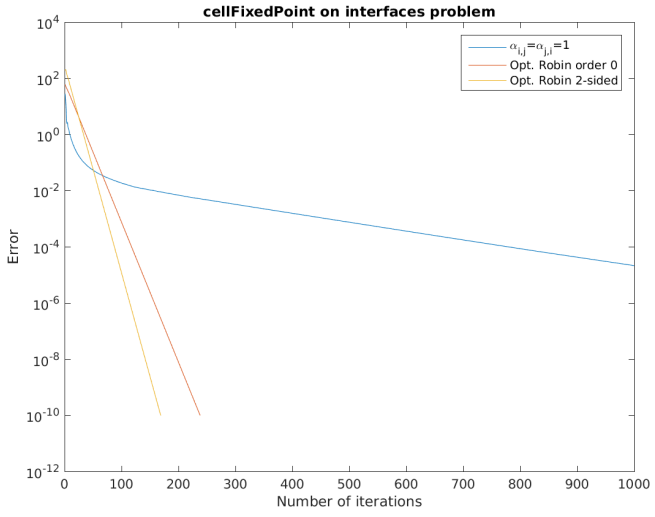



Figure 46: BVP sample 1 (nq = 2309), error

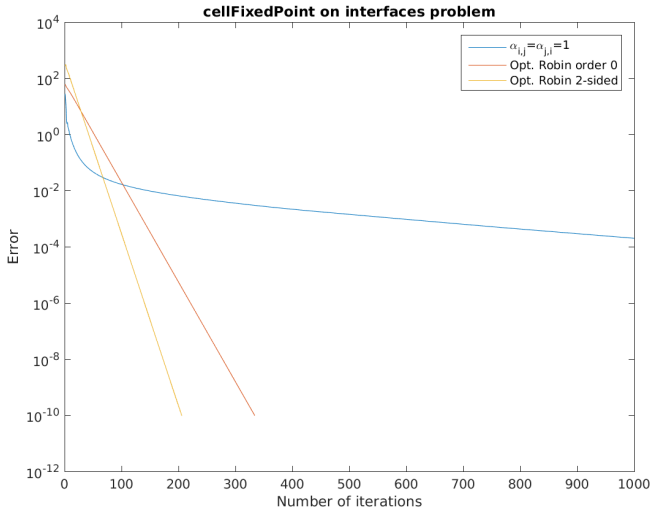



Figure 47: BVP sample 1, error with nq = 8663 ($N = 80$)

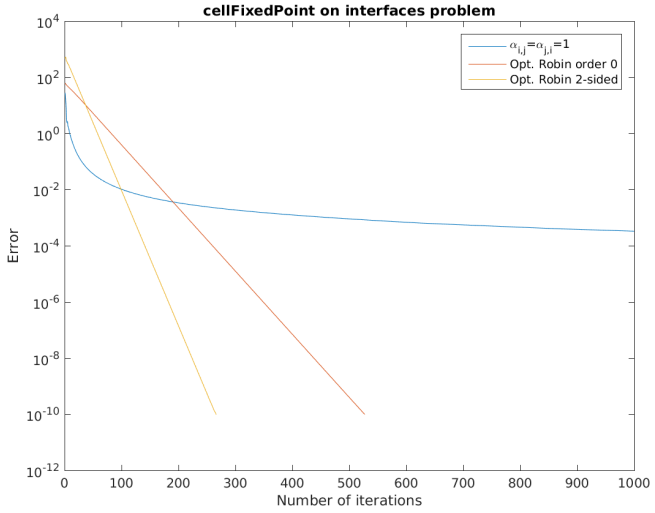



Figure 48: BVP sample 1, error with nq = 53474 ($N = 200$)

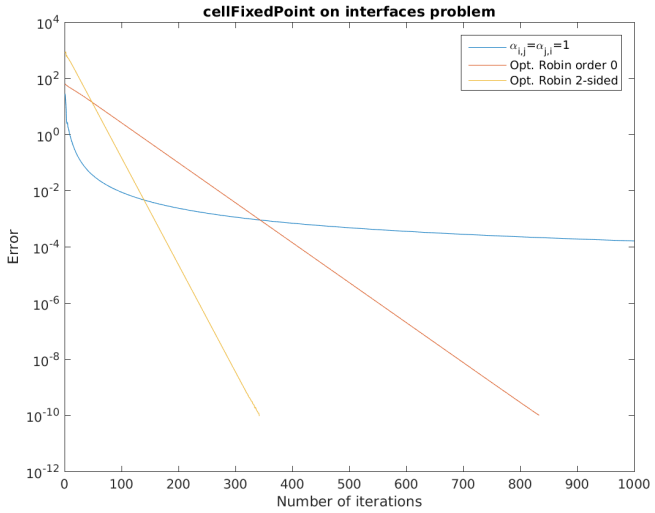



Figure 49: BVP sample 1, error with nq = 331984 ($N = 500$)

### B.1.2 BVP sample 1, 8-by-1 partitions

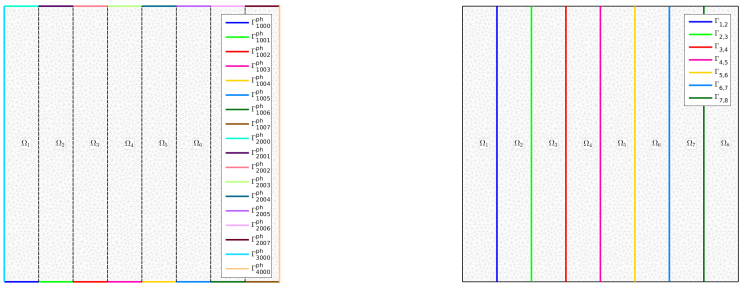


Figure 50: 2-by-3 partionned mesh (nq = 4995), boundaries (left) and interface (right)

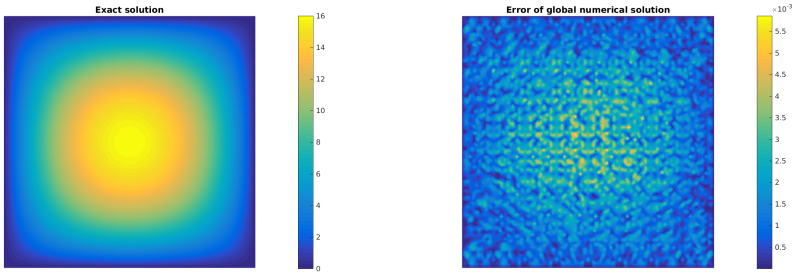



Figure 51: BVP sample 1 (nq = 4995), exact solution (left) and global numerical solution (right)

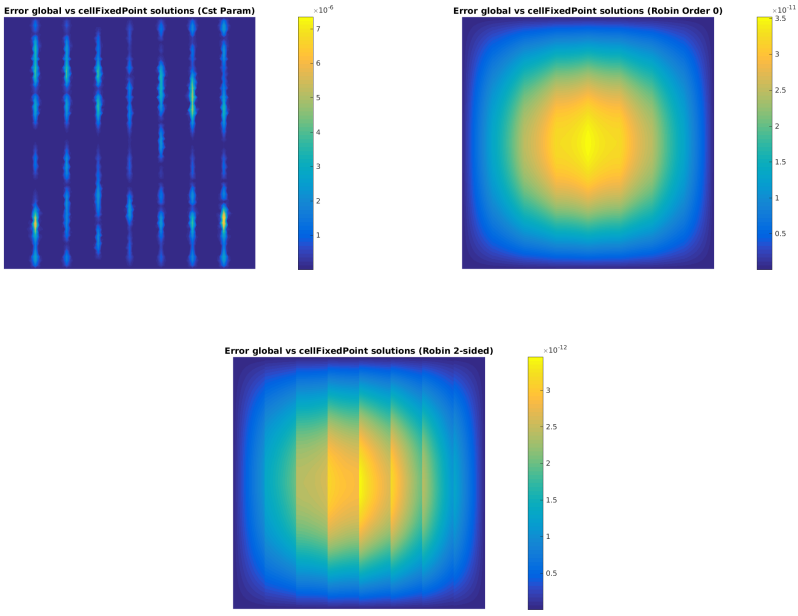



Figure 52: BVP sample 1 (nq = 4995), error between global and FPA numerical solution with Robin constant parameters $\alpha = 1$ (upper left), optimal Robin order 0 (upper right) and 2-sided optimal Robin (bottom)

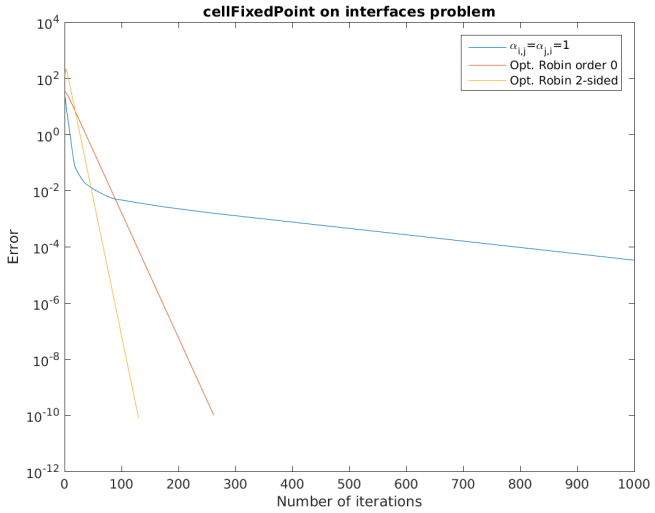



Figure 53: BVP sample 1 (nq = 4995), error

### B.1.3 BVP sample 1, 1-by-8 partitions

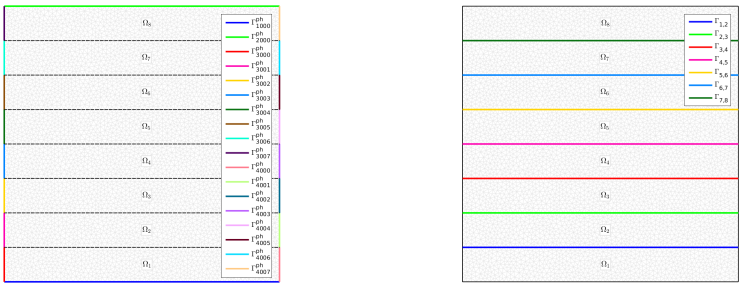


Figure 54: 2-by-3 partionned mesh (nq = 5011), boundaries (left) and interface (right)

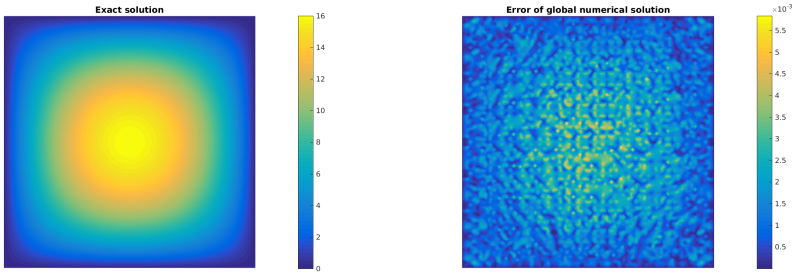



Figure 55: BVP sample 1 (nq = 5011), exact solution (left) and global numerical solution (right)

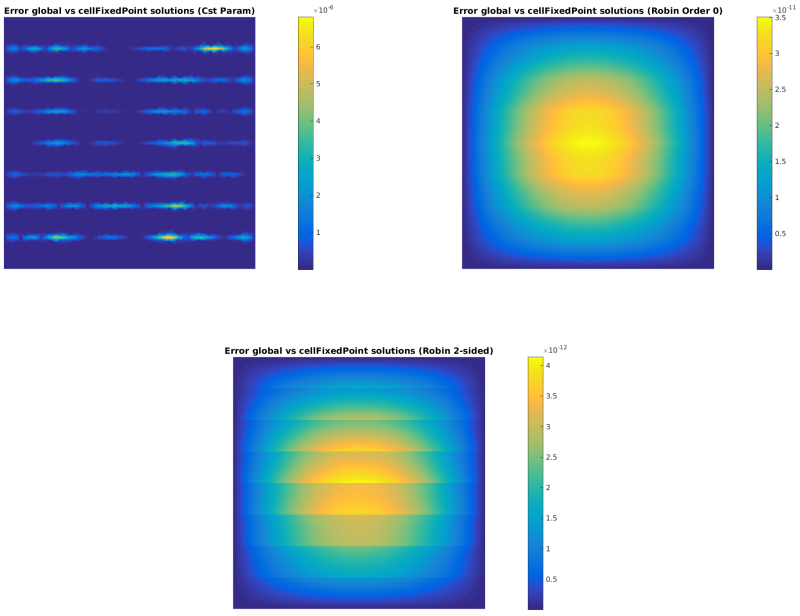



Figure 56: BVP sample 1 (nq = 5011), error between global and FPA numerical solution with Robin constant parameters $\alpha = 1$ (upper left), optimal Robin order 0 (upper right) and 2-sided optimal Robin (bottom)

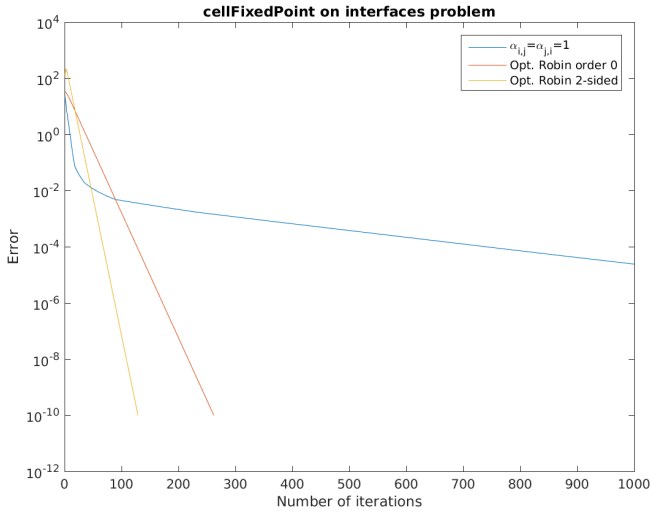



Figure 57: BVP sample 1 (nq = 5011), error

### B.1.4 BVP sample 1, 8-by-8 partitions

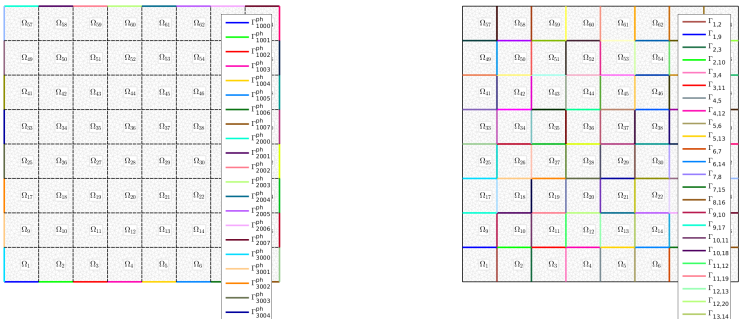


Figure 58: 8-by-8 partionned mesh (nq = 5931), boundaries (left) and interface (right)

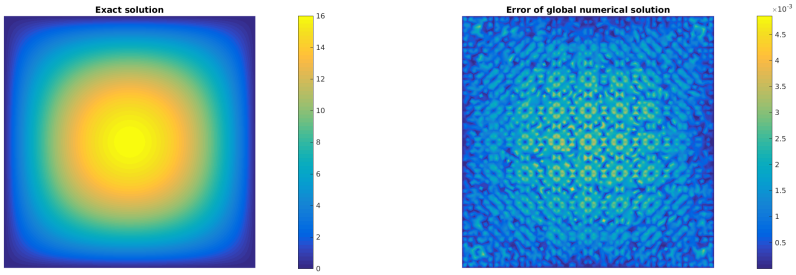



Figure 59: BVP sample 1 (nq = 5931), exact solution (left) and global numerical solution (right)

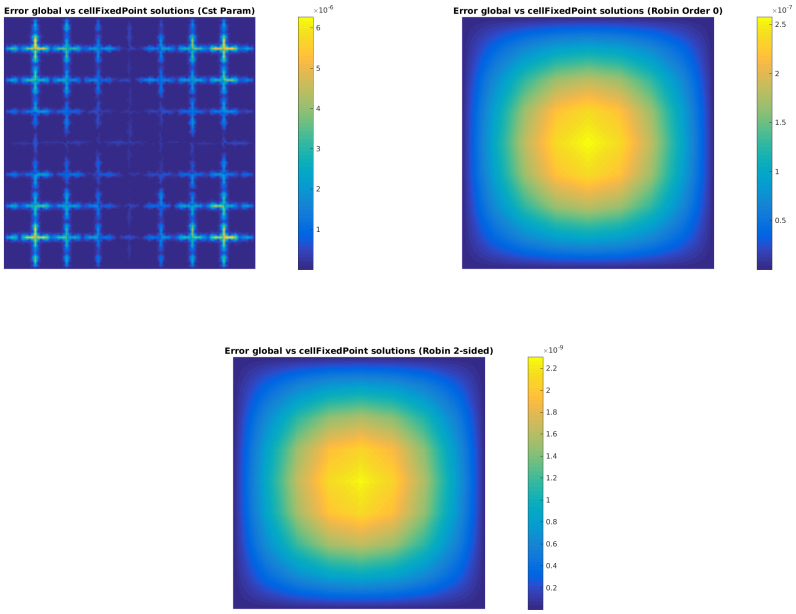



Figure 60: BVP sample 1 (nq = 5931), error between global and FPA numerical solution with Robin constant parameters $\alpha = 1$ (upper left), optimal Robin order 0 (upper right) and 2-sided optimal Robin (bottom)

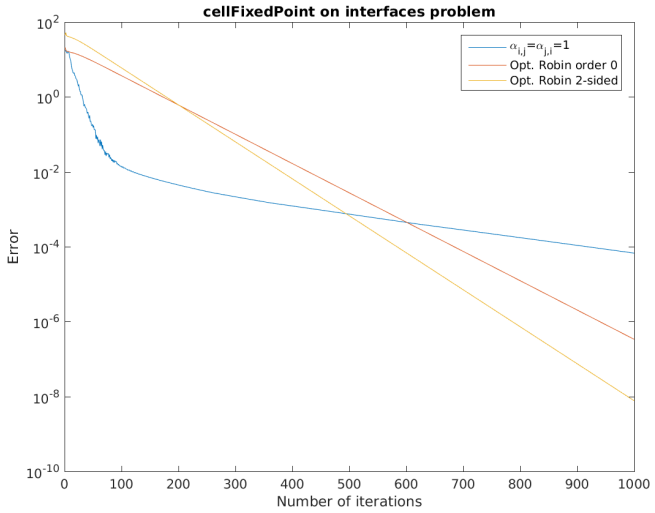



Figure 61: BVP sample 1 (nq = 5931), error

### B.1.5 BVP condenser, 15 partitions

# References

[1] Martin J Gander. Optimized schwarz methods. *SIAM Journal on Numerical Analysis*, 44(2):699–731, 2006.