Homogenization and COMSOL

IX Modelling Week UCM 2015

Master's in Mathematical Engineering

Participants: Carlos Arechalde Pérez
Pablo Cañones Martín
Denis Coccolo Góngora
Nadia Loy
Amarpreet Kaur

Instructor: David Gómez Castro

Madrid - July 6, 2015

Contents

1 Introduction

Homogenization is an important tool to solve complex problems composed of multiple elements with different properties. After homogenization of a system, a problem can be simplified to a single-element problem with a single effective value that takes into account the original heterogeneity of the problem. Such solution can be applied to many different Physical and Chemical problems.

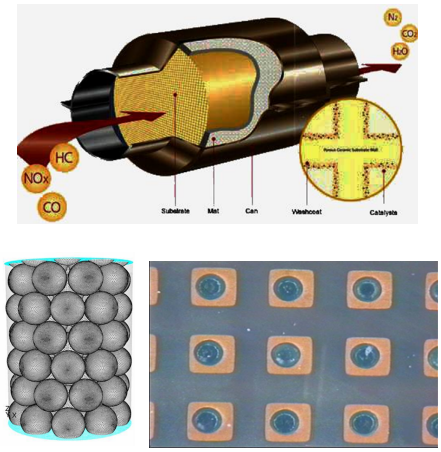


Figure 1: Example from Chemical Engineering (Exhaust pipe)

The importance of the homogenization is that it gives a tool to approximate our problem by one we can solve much more easily, and that gives us sufficient information. We pass to the limit in the way we don't study atoms individually, but rather the large structure properties.

Homogenization can be used for physical and chemical equations, such as Maxwell equations. From Maxwell equations, a single equation for Electric (E) and Magnetic (B) fields can be derived (Helmholtz equations) for the propagation of electromagnetic waves of frequency ω in vacuum, at speed of c.

$$\left(\nabla^2 + \frac{\omega^2}{c^2}\right)^2 E = 0 \tag{1}$$

The problem is that when waves propagate through a material, the electric and magnetic fields interact. Solving the equation taking into account the interaction of such a high number of point charges is simply impossible, and thus a spatial and temporal homogenization is made to define effective macroscopic properties like dielectric constant ε, magnetic permittivity μ and refraction index n, as well as new variables like Displacement D and Magnetic Induction H fields which accounts for the interaction of the electromagnetic fields in the whole medium of propagation. Nanotechnology has allowed us to artificially combine materials with these macroscopic properties. The propagation of waves through these new artificial mediums composed of different materials with different properties can also be simplified with a new homogenization process to yield new values for ε, μ and n, which in some cases can lead to exotic new phenomena like negative refraction or the apparition of bands where the propagation is not possible.

The possibility of design and fabrication of these new materials opens the path to the manipulation of light and its propagation. The cloaking problem gives rise to a Helmholtz equation after the homogenization of Maxwell's equations. As an example, let us consider the Figure 2.

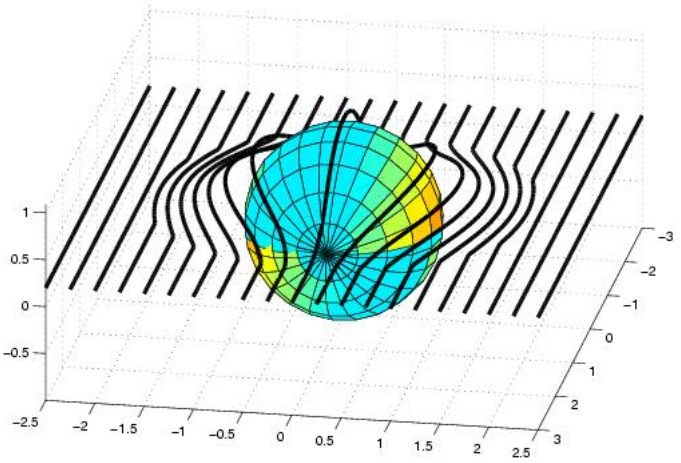


Figure 2: Microscopic level

What we see there is a small obstacle (microscopic level) and the lines, which represent the light. At this level the light bends around the obstacle. This property in a macroscopic level would produce cloaking (Figure 3).

Figure 3: Macroscopic level

To validate the homogenization we want for the Maxwell's equations in nanotechnology we work on a simplified model from Chemical Engineering. In our models we've focused on simple domains with regular obstacles 4.

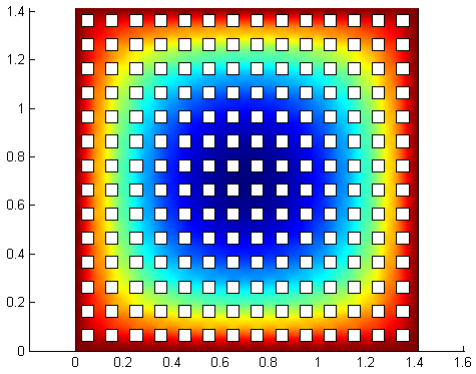


Figure 4: Example of a perforated domain

The purpose of this report is to study the convergence of solutions to the solution of the homogenized problem, compare with the theoretical results, study the computation complexity of the problem as the number of particles involves increases, and compare computation times with the homogenized problem.

For this problem we used COMSOL Multiphysics which will be introduced in 3.

The study steps were:

1. Compute complete model solution u^ε

$$\begin{cases} -\Delta u_\varepsilon = f & \text{in } \Omega^\varepsilon, \\ \frac{\partial u_\varepsilon}{\partial \nu} + \varepsilon g(u^\varepsilon) = 0 & \text{on } S^\varepsilon, \\ u_\varepsilon = 1 & \text{on } \partial\Omega, \end{cases}$$

2. For homogenizing, we solve the cell problem χ_i

$$\begin{cases} -\Delta \chi_i = 0 & \text{in } Y \setminus T, \\ \frac{\partial(\chi_i + y_i)}{\partial \nu} = 0 & \text{on } \partial T, \\ \chi_i & Y\text{-periodic.} \end{cases}$$

3. Then we compute the $a_0(T)$ matrix.

$$a_0(T)_{ij} = q_{ij} = \delta_{ij} + \frac{1}{|Y \setminus T|} \int_{Y \setminus T} \frac{\partial \chi_j}{\partial y_i} dy,$$

4. And finally we can compute homogenized problem.

$$\begin{cases} -\operatorname{div}(a_0(T)\nabla u) + \frac{|\partial T|}{|Y \setminus T|} g(u) = f & \text{in } \Omega, \\ u = 1 & \text{on } \partial\Omega. \end{cases}$$

2 Mathematical Model

The equation that we want to study is as follows:

$$\begin{cases} -\Delta u_\varepsilon = f & \text{in } \Omega^\varepsilon, \\ \frac{\partial u_\varepsilon}{\partial \nu} + \varepsilon^{-\beta} g(u^\varepsilon) = 0 & \text{on } S^\varepsilon, \\ u_\varepsilon = 1 & \text{on } \partial\Omega, \end{cases} \tag{2}$$

where we have a laplace equation inside the domain (Ω^ε) and two types of boundary conditions, the exterior boundary ($\partial\Omega$) and the boundary of the obstacles (S^ε).

The inside pellets are defined as:

$$\begin{aligned} T_k^\varepsilon &= (\varepsilon k + \varepsilon^\alpha T) & k &\in \mathbb{Z}^n \\ T^\varepsilon &= \bigcup_{\overline{T_k^\varepsilon} \subset \Omega} T_k^\varepsilon & \Omega^\varepsilon &= \Omega \setminus \overline{T^\varepsilon} \\ \partial\Omega^\varepsilon &= S^\varepsilon \cup \partial\Omega, & S^\varepsilon &= \bigcup_{\overline{T_k^\varepsilon} \subset \Omega} \partial T_k^\varepsilon \end{aligned}$$

For the solution to be stable and converge to the homogenized system we have to choose: $\beta = \alpha(N-1) - N$

In the following figure we can see the solution given by COMSOL to this system.

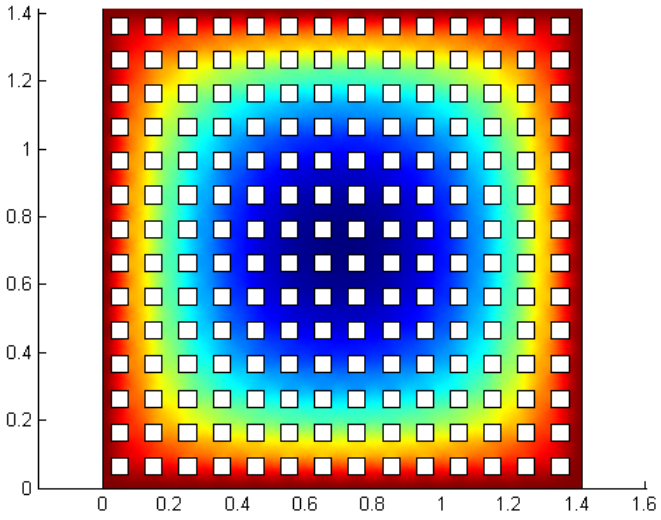
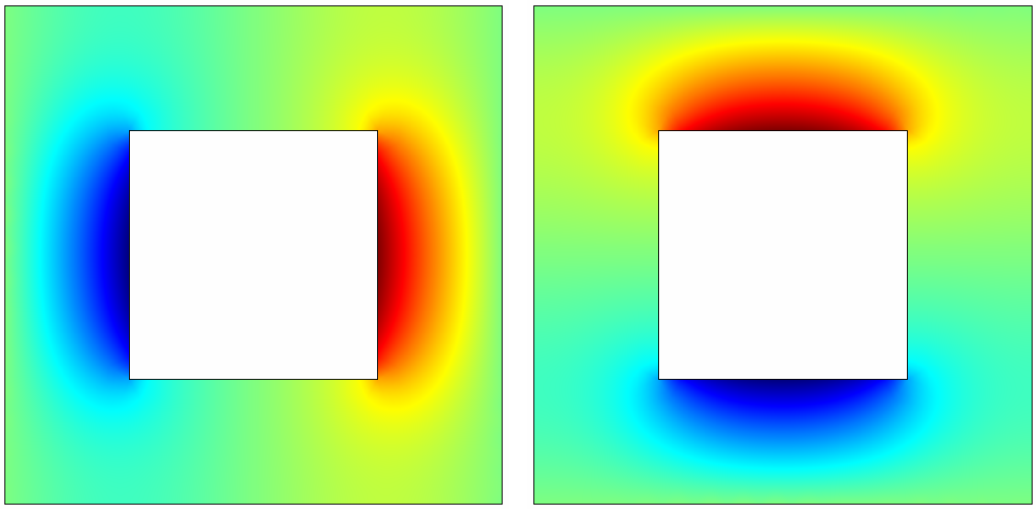


Figure 5: Perforated domain with $\alpha = \beta = 1$

To study the relation between, the problem with obstacles to the homogenized one we define the effectiveness of the system η_ε, a measure used in chemical engineering:

$$\eta_\varepsilon = \frac{1}{|S_\varepsilon|} \int_{S_\varepsilon} g(u_\varepsilon) dS$$

This is the original problem we are facing and now we would like to obtain a homogenized system that takes into account the obstacles without having to solve them explicitly. To do this we'll use theorem 1.



(a) χ_1 solution (b) χ_2 solution

Figure 6: Solutions of (4) for T a square

Theorem 1 (Conca, Díaz, Timofte, Liñán[1]) *Let $\alpha = \beta = 1$. Under weak regularity conditions on g there exists $P^\varepsilon u^\varepsilon$ extension of u^ε such that*

$$P^\varepsilon u^\varepsilon \overset{H^1}{\rightharpoonup} u$$

u is the unique solution of the following homogenized problem

$$\begin{cases} -\operatorname{div}(a_0(T)\nabla u) + \frac{|\partial T|}{|Y\setminus T|} g(u) = f & \text{in } \Omega, \\ u = 1 & \text{on } \partial\Omega. \end{cases} \tag{3}$$

This theorem gives us a way to solve the homogenized problem. Our next problem is obtaining the value of $a_0(T)$, the element that contains the effect of the boundary of the obstacles.

Let us write the matrix $a_0(T) = (q_{ij})$, then

$$q_{ij} = \delta_{ij} + \frac{1}{|Y \setminus T|} \int_{Y\setminus T} \frac{\partial \chi_j}{\partial y_i} dy,$$

where χ_i are the solutions of the so-called **cell problems**:

$$\begin{cases} -\Delta \chi_i = 0 & \text{in } Y \setminus T, \\ \frac{\partial(\chi_i + y_i)}{\partial \nu} = 0 & \text{on } \partial T, \\ \chi_i & Y\text{-periodic.} \end{cases} \tag{4}$$

In our case we have to solve two problems, χ_1 and χ_2, for the two dimensions.

Once we obtain the solution of the cell problems we can solve the homogenized one. The last thing to check is the convergence of the effectiveness of the problem with obstacles to the homogenized problem

Definition 1 *We define the effectiveness of the homogenized problem as η.*

$$\eta = \frac{1}{|\Omega|} \int_\Omega g(u) dx \tag{5}$$

Recalling the formula for the effectiveness of the problem with obstacles:

$$\eta_\varepsilon = \frac{1}{|S_\varepsilon|} \int_{S_\varepsilon} g(u_\varepsilon) dS,$$

there exists the following result:

Proposition 1 *For $\varepsilon \to 0$, it follows that $\eta_\varepsilon \to \eta$.*

Which means that solving the problem with infinitely small obstacles is equivalent to solving the homogenized problem.

3 Methodology

For the resolution of the differential equations we used COMSOL Multiphisics, a finite elements solver for physics and engineering applications. It allows to create a domain of any kind, include various types of boundary conditions and one or more differential equations. Also, using the LiveLink tool, it allows to create a Matlab code that we have used to generalize the construction of the obstacles in our domains.

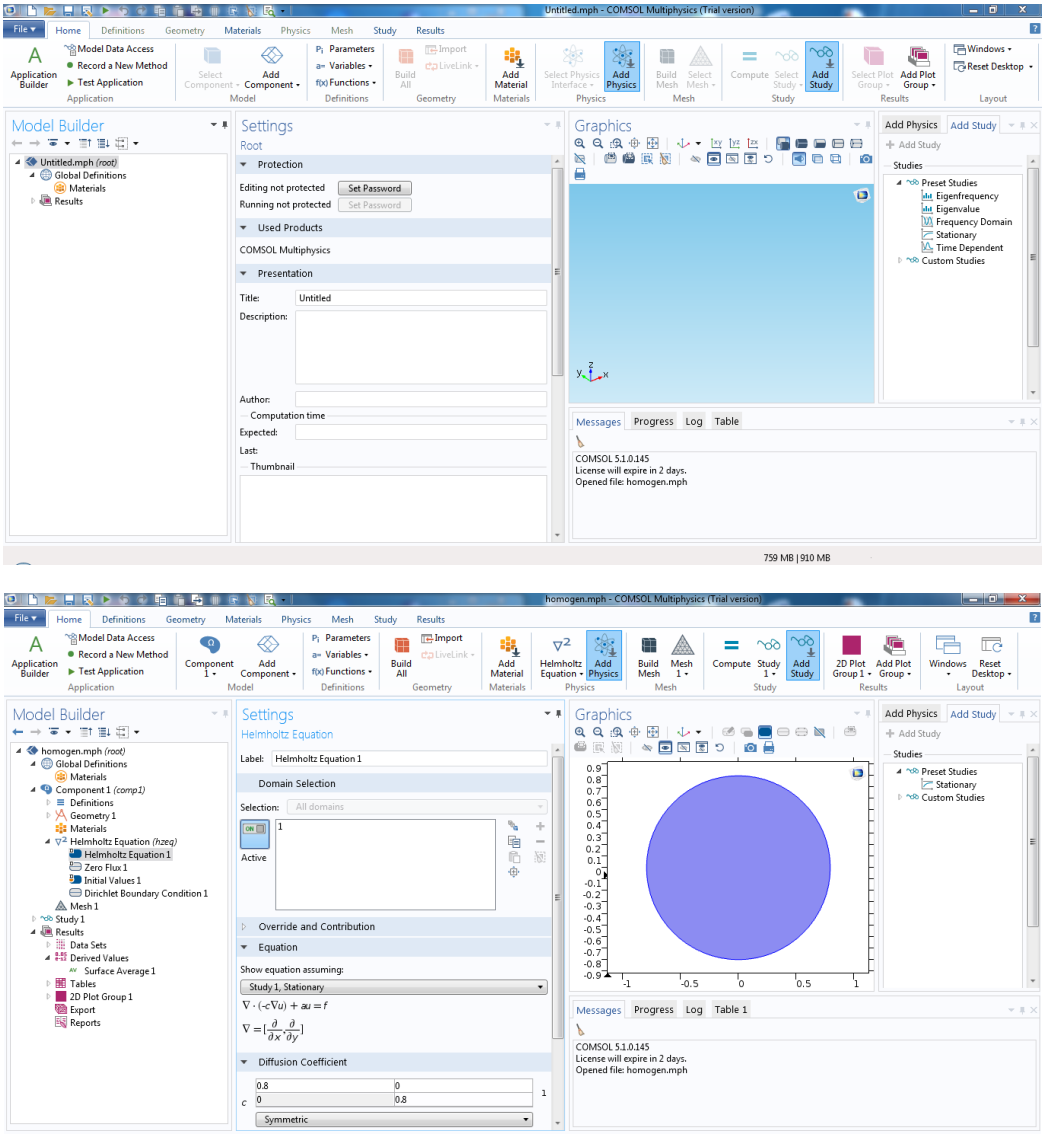


Figure 7: COMSOL Interface

3.1 Domains

Each of us decided on a shape for the domain and another for the included obstacles. This is an example of the loop used to include all obstacles (rectangles of certain dimensions) that fit into an ellipse:

```
sdifference = ...
    'model.geom(''geom1'').feature(''dif1'').selection(''input2'').set({';
for j = 1 : 2*semiax(2)/eps + 1
  for i = 1 : 2*semiax(1)/eps + 1}
    l_down = [-semiax(1) + (i-1)*eps + (eps - lados(1))/2,
              -semiax(2) + (j-1)*eps + (eps - lados(2))/2];
    r_down = l_down + [lados(1) 0];
    l_up = l_down + [0 lados(2)];
    r_up = l_down + lados;
    if in_Elipse(l_down, a, b) && in_Elipse(r_down, a, b) &&
     in_Elipse(l_up, a, b) && in_Elipse(r_up, a, b)
       nRec = nRec + 1;
       nombR = ['r' num2str(nRec)];
       model.geom('geom1').create(nombR, 'Rectangle');
       model.geom('geom1').feature(nombR).set('size', { num2str(lados) });
       model.geom('geom1').feature(nombR).set('pos', { num2str(l_down) });
       sdifference = [sdifference ' ''', nombR, ''' '];
    end
  end
end
sdifference = [sdifference, '});'];
```

Later, we can build the difference of Ω and the pellets:

```
model.geom('geom1').create('dif1', 'Difference');
model.geom('geom1').feature('dif1').selection('input').set({'e1'});
eval(sdifference);
```

We have simulated that each pellet is inside a periodicity cell. The input parameter of the function is ϵ, the side of this periodicity cell, that has four times the area of the pellet. Figures 8-12 show what happens if we change the value of ϵ

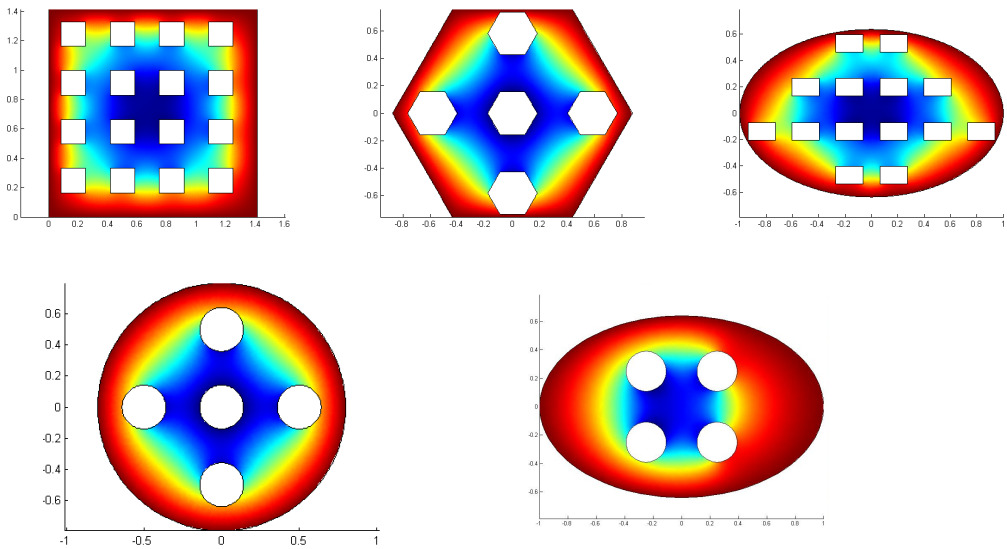


Figure 8: Pictures for $\epsilon = \frac{1}{3}$

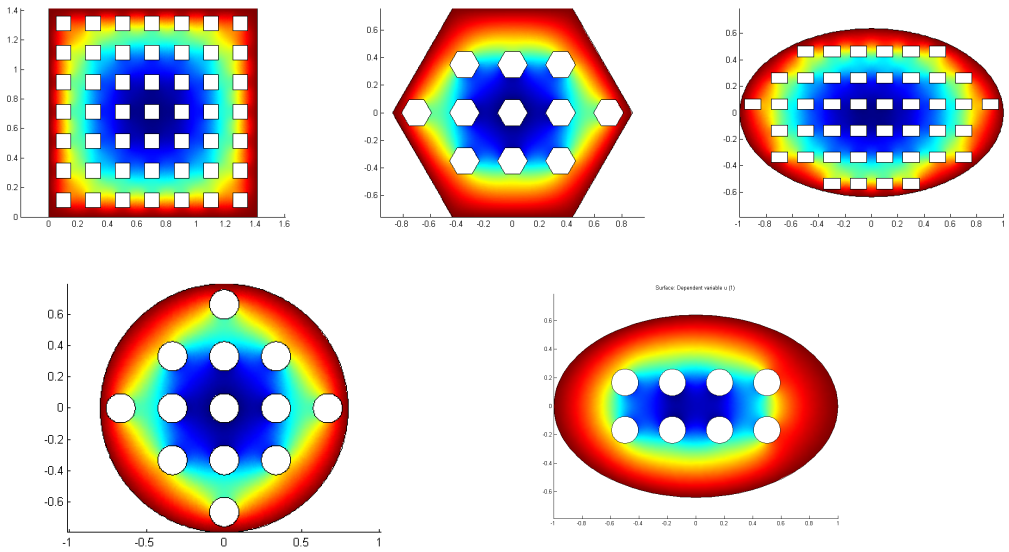


Figure 9: Pictures for $\epsilon = \frac{1}{5}$

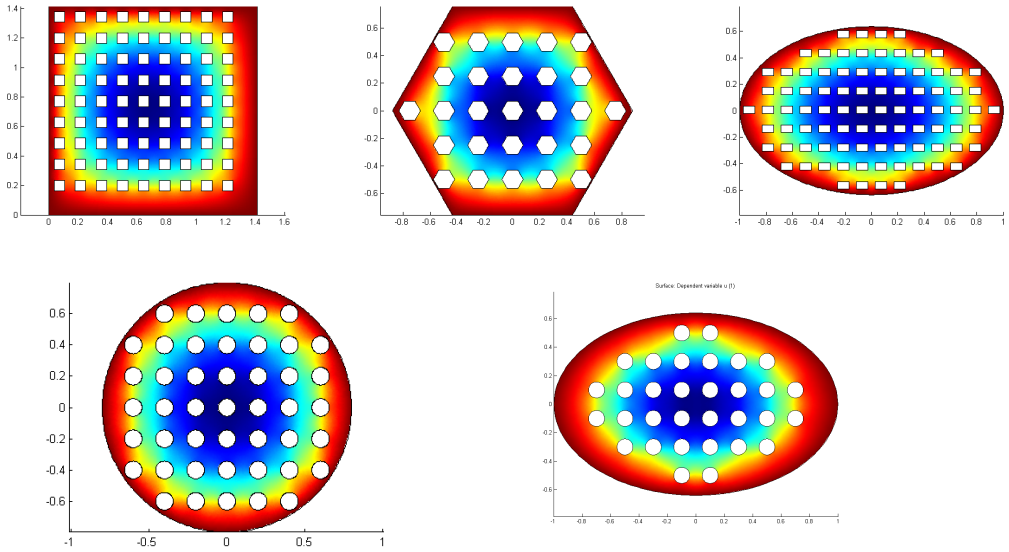


Figure 10: Pictures for $\epsilon = \frac{1}{7}$

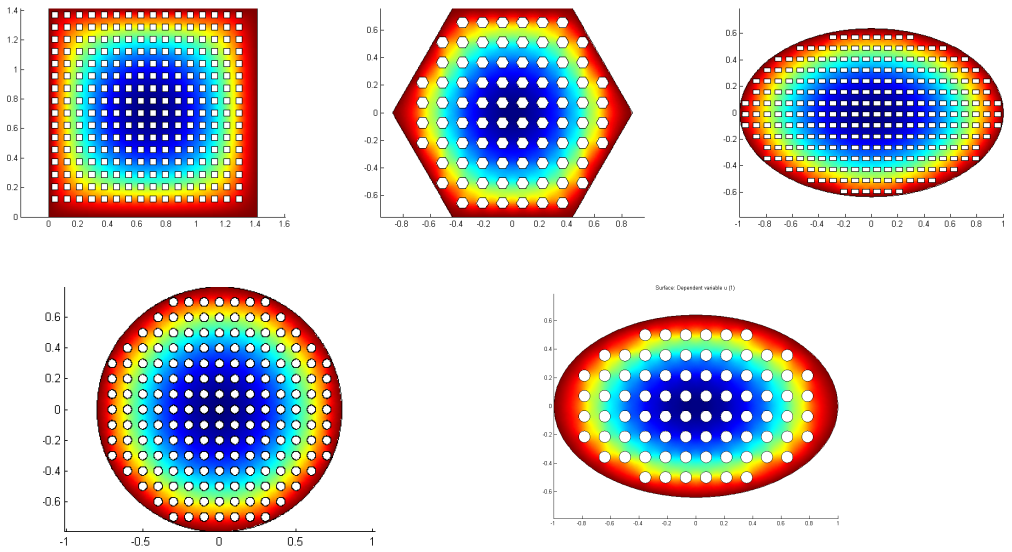


Figure 11: Pictures for $\epsilon = \frac{1}{12}$

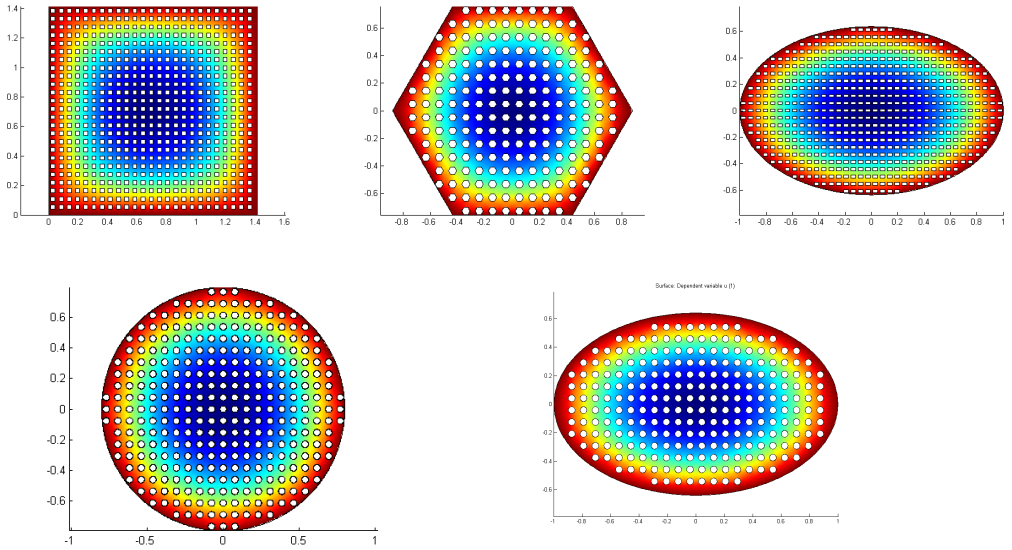


Figure 12: Pictures for $\epsilon = \frac{1}{18}$

From all of them, the most interesting results are obtained for the smallest ϵ, because we can see that diffusion in this case is pretty similar to the homogenized problem, as we expected, because of the Theorem.

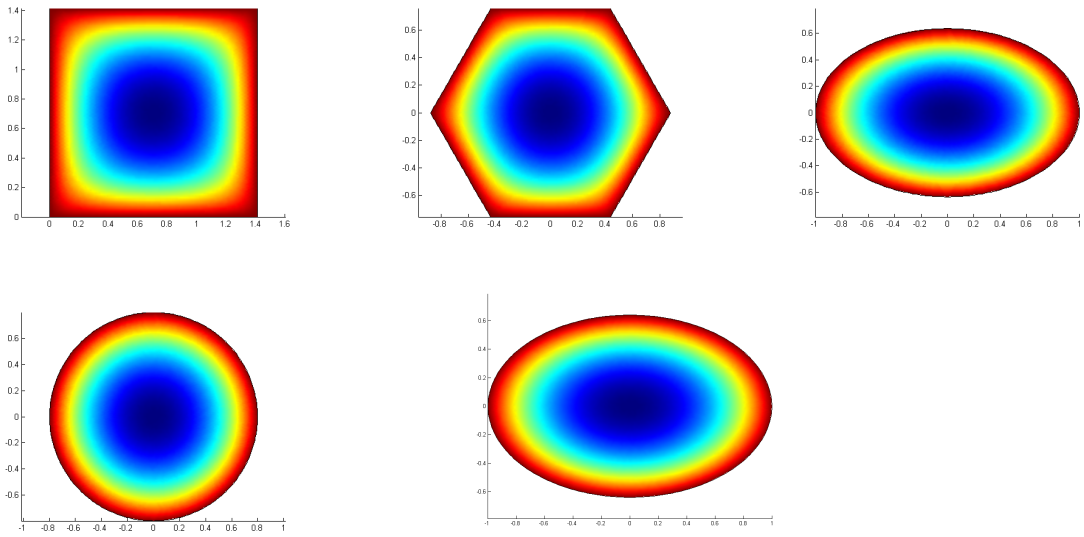


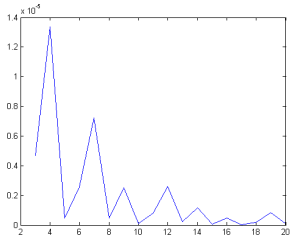
Figure 13: Solutions of the homogenized problems

4 Results

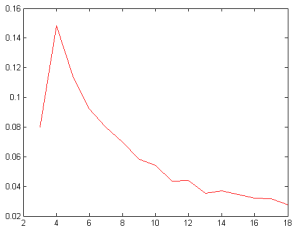
In order to study the convergence from the problem with obstacles to the homogenized one we encounter a problem: the solutions are not defined in the same domain, the first one has holes inside that the homogenized one ignores. To be able to compute a difference between both we interpolate the solution inside the holes, it is a very rough approximation to avoid the problem but we were able to obtain acceptable results of convergence.

4.1 L_2 norm

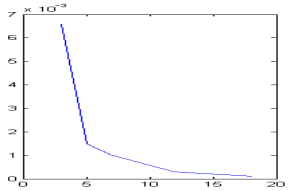
Following the main theorem that proves a result of convergence using the L_2 norm we compute the difference between the two problems using it. As the theorem said, we obtain that, as ε goes to zero so does the difference.



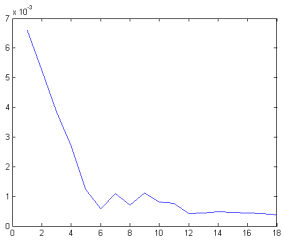
(a) Squares inside square



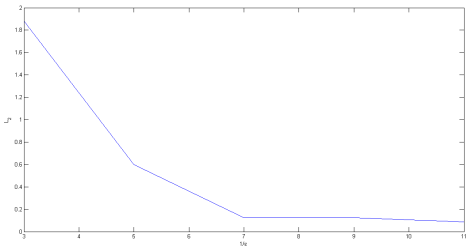
(b) Hexagons inside hexagon



(c) Rectangles inside ellipse



(d) Circles inside circle

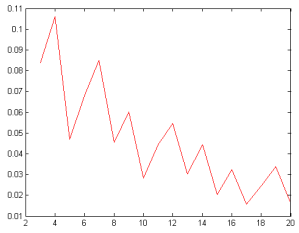


(e) Circles inside ellipse

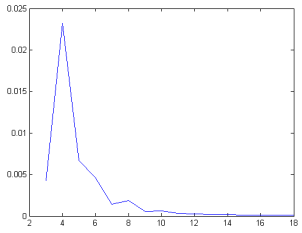
Figure 14: L_2 norm convergence

4.2 L_∞ norm

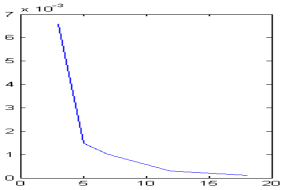
Since we already had a way to compute the difference between both solutions we went ahead and computed the L_∞ norm which also shows that there may be convergence to the homogenized problem. This is something that wasn't theoreticaly proved in the main theorem.



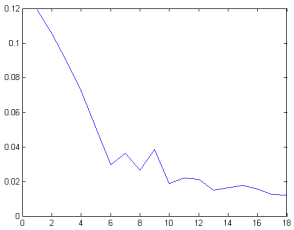
(a) Squares inside square



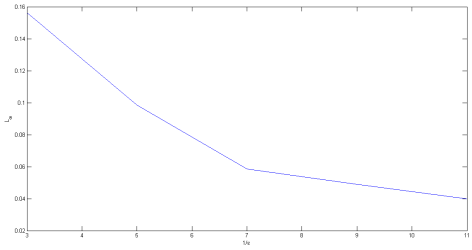
(b) Hexagons inside hexagon



(c) Rectangles inside ellipse



(d) Circles inside circle

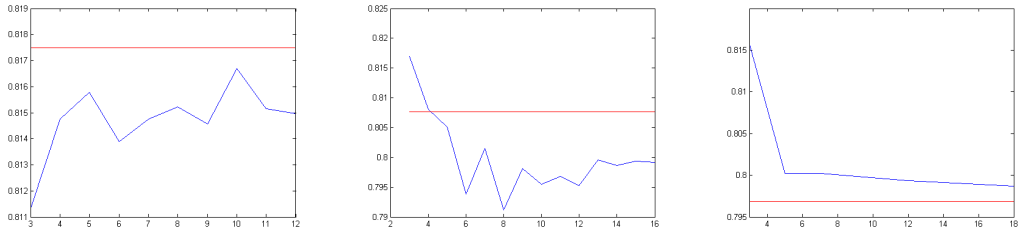


(e) Circles inside ellipse

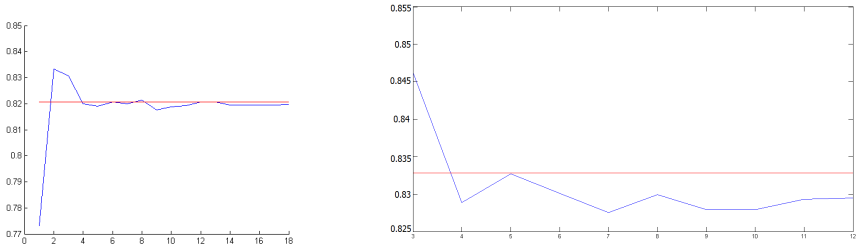
Figure 15: L_∞ norm convergence

4.3 Convergence efectiveness

Finally we take a look at the measure of efectiveness, computed in the problems with obstacles and in the homogenized one. Here it's clear that this measure behaves better depending on the geometry of the domains and graphics but in all five cases the difference is really small (see range of the vertical axis in any of them).



(a) Squares inside square (b) Hexagons inside hexagon (c) Rectangles inside ellipse



(d) Circles inside circle (e) Circles inside ellipse

Figure 16: Convergence efectiveness result: Red line shows the value of non homogeneous problem. Blue line shows the convergence of the homogeneous problem as a function of the value $n = \frac{1}{\varepsilon}$.

4.4 Computational complexity

As was expected theoretically, the smaller ε gets, the harder it is to compute the solution of the problem with obstacles and hence the usefulness of the homogenized solution. In the following graphs we can see the increase in points in the finite elements mesh and the computational time.

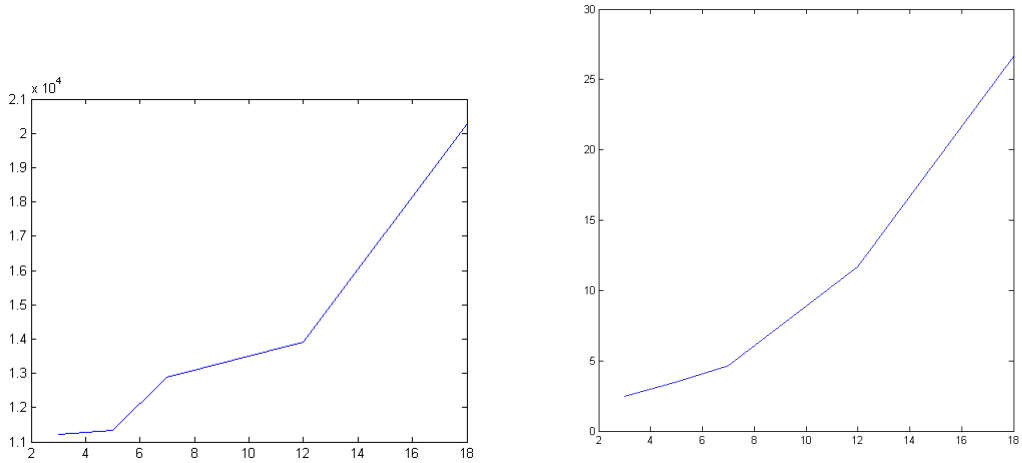


Figure 17: Left: Number of points against $1/\varepsilon$. Right: Computation time in seconds against $1/\varepsilon$

5 Conclusions

5.1 What we have learned

- The first, and quite important, thing to conclude from this work in the Modelling Week is the powerful help COMSOL is for solving numerical problems like the one we had to do. Since the creation of the mesh is automatic once it gets the domain and the user doesn't have to write any code concerning the finite elements resolution, we can save a lot of time in programing. Also, since the LiveLink allows us to export the COMSOL program as a Matlab code, we were able to write a sort of simple code to create problems with different sizes of obstacles and obtain the graphics for the effectiveness.

- Concerning the actual problem we were studying, we can conclude that solving the homogenized problem simplifies the computation since we don't have to construct the obstacles inside the domain and define the boundary conditions on them. In the limit case with a large number of really small obstacles, changing to the homogenized version really improves performance. See Figure 17

- By using the homogenized problem we can study in a macroscopic scale the microscopic properties the obstacles introduce when ε tends to zero without, as we said before, the computational complexity growth.

- Another thing we have seen is that the convergence from the problem with obstacles to the homogenized one when the size of the obstacles diminishes is quite fast. The L_2 norm behaviour between the original and the homogenized problems varies with the different domains and obstacles we used but in general we obtained clear conclusions of convergence with few values of ε and in acceptable time. See Figure 14.

- Finally, we tried to compute the convergence of the solution to the homogenized one in the L_∞ norm, something that was not proved in the main theorem that gave us the equation for the homogenized problem, and also got suggestions that there may exists convergence in this norm too. See Figure 15

5.2 Future work

There are many other applications where the use of a homogenized solution can solve a lot of computational problems like the ones we speak about here. For example:

- Study the homogenization in the Maxwell equation. Here we used a simple Laplace equation one but doing the same with the Maxwell one could eventually lead to the cloacking effect.

- Study non periodic domains. COMSOL and LiveLink have been of great help since the domains and obstacles we used were simple ones where the equations for $a_0(T)$ were easy to solve. In more general domains with complex geometries this step would require more computations.

- Apply the homogenization to the Navier Stokes equations for porous medium. In this case we would have the similar situation with a domain filled with obstacles where the solution behaves differently.

- Study the convergence of the solution in higher dimensions to either verify or disprove the idea of convergence.

References

[1] C.Conca, J.I. Díaz, A. Liñán and C.Timofte. Homogenization in Chemical Reactive Flows. *Electronic Journal of Differential Equations* 40 (2004), pp 1-22.

[2] J.-L Auriault, C.Boutin and C. Geindreau. Homogenization of Coupled Phenomena in Heterogeneous Media. Paris: ISTE and John Wiley & Sons, 2009.

A Example of a COMSOL code

Code to build the domain and the obstacles:

```
%Factor so that the big hexagon has volume 1
scale=sqrt(4/(sqrt(3)*3));
model.geom('geom1').create('b1', 'BezierPolygon');
x0=[-0.5 0.5 1 0.5 -0.5 -1]*scale;
y0=[-sqrt(3)/2 -sqrt(3)/2 0 sqrt(3)/2 sqrt(3)/2 0]*scale;
x=num2cell(x0);
x=cellfun(@num2str,x,'UniformOutput',0);
y=num2cell(y0);
y=cellfun(@num2str,y,'UniformOutput',0);
model.geom('geom1').feature('b1').set('p', [x;y]);

epsil=2/N; %size of the small squares
%factor so that the size of the small hexagon is 1/4 of epsil^2
k=1/sqrt(6*sqrt(3));
d1=[1,0]*epsil;
d2=[0,1]*epsil;
cor=[-1 -1]+epsil/2;
sidesl1=0;
sidesl2=0;
sidesr1=0;
sidesr2=0;
hexas=1;

i=0 ;
while i<N+1
j=0;
while j<N+1
    %Compute coordinates with respect to some big hexagon
    %then multiply times epsilon and k
    x=(cor(1)+d1(1)*i+d2(1)*j)+[-0.5 0.5 1 0.5 -0.5 -1]*epsil*k;
    y=(cor(2)+d1(2)*i+d2(2)*j)+[-sqrt(3)/2 -sqrt(3)/2 0 sqrt(3)/2 ...
        sqrt(3)/2 0]*epsil*k;
    %Check feasibility
    if ¬feasible(x,y)
        j=j+1;
        continue
    end
    model.geom('geom1').create(['b',int2str(hexas+1)], 'BezierPolygon');

    %Check how many sides to add (may need revission)
    if abs(x(3)+0.5)<epsil^10
        sidesl1=sidesl1+6;
        sidesl2=sidesl2+6;
    elseif x(3)<-0.5
        sidesl1=sidesl1+6;
        sidesl2=sidesl2+6;

    elseif x(2)≤-0.5
        sidesl1=sidesl1+6;
        sidesl2=sidesl2+6;
    elseif x(1)≤-0.5
        sidesl1=sidesl1+4;
        sidesl2=sidesl2+4;

    elseif abs(x(6)+0.5)<epsil^10
```

```
        sidesl2=sidesl2+2;
    elseif x(6)<-0.5
        sidesl1=sidesl1+2;
        sidesl2=sidesl2+2;
    end

    if abs(x(6)-0.5)<epsil^10
        sidesr1=sidesr1+6;
        sidesr2=sidesr2+4;
    elseif x(6)>0.5
        sidesr1=sidesr1+6;
        sidesr2=sidesr2+6;

    elseif x(1)>0.5
        sidesr1=sidesr1+4;
        sidesr2=sidesr2+4;

    elseif x(2)>0.5
        sidesr1=sidesr1+2;
        sidesr2=sidesr2+2;
    end

    %Once the hexagon is accepted, scale it to the actual size of the big
    %one
    x=x*scale;
    y=y*scale;
    x=num2cell(x);
    x=cellfun(@num2str,x,'UniformOutput',0);
    y=num2cell(y);
    y=cellfun(@num2str,y,'UniformOutput',0);
    model.geom('geom1').feature(['b',int2str(hexas+1)]).set('p', [x;y]);
    others{hexas}=['b',int2str(hexas+1)];
    hexas=hexas+1;
    j=j+1;
end
i=i+1;
end
model.geom('geom1').create('dif1', 'Difference');
model.geom('geom1').feature('dif1').selection('input2').set(others);
model.geom('geom1').feature('dif1').selection('input').set({'b1'});
model.geom('geom1').run;
```

Function to check if an obstacle is inside the domain:

```
function f=feasible(x,y)
C=[[     0        1 ];
    [-sqrt(3)/2  0.5];
    [-sqrt(3)/2 -0.5];
    [    0       -1];
    [-sqrt(3)/2  0.5];
    [ sqrt(3)/2  0.5]];
b=[-sqrt(3)/2 -sqrt(3)/2 -sqrt(3)/2 -sqrt(3)/2 sqrt(3)/2 -sqrt(3)/2];
%sign for the inequality
sign=[-1 -1 -1 -1 1 -1];
f=1;
for i=1:6
    if (C(i,:)*[x(i);y(i)]-b(i))*sign(i)≥0
        f=0;
        return
```

```
    end
end
```

B Code to solve the homogeneous case

```
function out = Ellipsis_andRect_converg
%
% Ellipsis_andRect_converg.m
%
% Model exported on Jun 18 2015, 11:20 by COMSOL 5.1.0.145.

import com.comsol.model.*
import com.comsol.model.util.*

model = ModelUtil.create('Model');

model.modelPath('C:\Users\usupclab-47\Desktop\MW');

model.label('Ellipsis_andRect_converg.mph');

model.comments(['Untitled\n\n']);

model.modelNode.create('comp2');

model.file.clear;

model.geom.create('geom2', 2);

model.mesh.create('mesh2', 'geom2');

model.geom('geom2').create('e1', 'Ellipse');
model.geom('geom2').feature('e1').set('pos', {'0' '0'});
model.geom('geom2').feature('e1').set('semiaxes', {'1' '2/pi'});
model.geom('geom2').run;

model.view('view1').tag('view2');

model.coordSystem('sys1').tag('sys2');

model.physics.create('hzeq', 'HelmholtzEquation', 'geom2');
model.physics('hzeq').create('dir1', 'DirichletBoundary', 1);
model.physics('hzeq').feature('dir1').selection.set([1 2 3 4]);

model.frame('material1').tag('material2');

model.result.table.create('tbl1', 'Table');
model.result.table.create('tbl2', 'Table');
model.result.table.create('tbl3', 'Table');

model.view('view2').label('View 2');
model.view('view2').axis.set('abstractviewxscale', ...
    '0.0066006602719426155');
model.view('view2').axis.set('ymin', '-1.0033003091812134');
model.view('view2').axis.set('xmax', '1.2937294244766235');
model.view('view2').axis.set('abstractviewyscale', ...
    '0.0066006602719426155');
```

```
model.view('view2').axis.set('abstractviewbratio', '-0.2853981852531433');
model.view('view2').axis.set('abstractviewtratio', '0.2853981852531433');
model.view('view2').axis.set('abstractviewrratio', '0.264026403427124');
model.view('view2').axis.set('xmin', '-1.2937294244766235');
model.view('view2').axis.set('abstractviewlratio', '-0.264026403427124');
model.view('view2').axis.set('ymax', '1.0033003091812134');

model.coordSystem('sys2').label('Boundary System 2');
model.coordSystem('sys2').set('name', 'sys2');

model.physics('hzeq').feature('heq1').set('c', {'1-0.1403' '0' '0' ...
    '1-0.35038'});
model.physics('hzeq').feature('heq1').set('a', '2.05*4/3');
model.physics('hzeq').feature('heq1').set('f', '0');
model.physics('hzeq').feature('dir1').set('r', '1');

model.mesh('mesh2').run;

model.result.table('tbl1').comments('Line Average 1 (u)');
model.result.table('tbl2').comments('Surface Average 2 (u)');
model.result.table('tbl3').comments('Surface Integration 1 (0.6911-u2)');

model.study.create('std2');
model.study('std2').create('stat', 'Stationary');

model.sol.create('sol2');
model.sol('sol2').study('std2');
model.sol('sol2').attach('std2');
model.sol('sol2').create('st1', 'StudyStep');
model.sol('sol2').create('v1', 'Variables');
model.sol('sol2').create('s1', 'Stationary');
model.sol('sol2').feature('s1').create('fc1', 'FullyCoupled');
model.sol('sol2').feature('s1').feature.remove('fcDef');

model.study('std2').feature('stat').set('initstudyhide', 'on');
model.study('std2').feature('stat').set('initsolhide', 'on');
model.study('std2').feature('stat').set('solnumhide', 'on');
model.study('std2').feature('stat').set('notstudyhide', 'on');
model.study('std2').feature('stat').set('notsolhide', 'on');
model.study('std2').feature('stat').set('notsolnumhide', 'on');

model.result.dataset.create('dset2', 'Solution');
model.result.dataset.create('dset3', 'Solution');
model.result.dataset.remove('dset1');
model.result.numerical.create('av2', 'AvSurface');
model.result.numerical('av2').set('data', 'dset3');
model.result.numerical('av2').selection.set([1]);
model.result.numerical('av2').set('probetag', 'none');
model.result.create('pg1', 'PlotGroup2D');
model.result('pg1').set('data', 'dset3');
model.result('pg1').create('surf1', 'Surface');

model.study('std2').label('Study 1');

model.sol('sol2').attach('std2');
model.sol('sol2').runAll;

model.result.numerical('av2').set('table', 'tbl2');
model.result.numerical('av2').setResult;

```

```
109 out = model;
```

C Code to solve the non homogeneous case

```
1  function [out1, out2] = Ellipsis_andRect_addAve(eps)
2  %
3  % Ellipsis_andRect_notScale.m
4  %
5  % Model exported on Jun 16 2015, 13:50 by COMSOL 5.1.0.145.
6
7  import com.comsol.model.*
8  import com.comsol.model.util.*
9
10 model = ModelUtil.create('Model');
11
12 model.modelPath('C:\Users\usupclab-47\Desktop\MW');
13
14 model.label('Ellipsis_andRect_notScale.mph');
15
16 model.comments(['Untitled\n\n']);
17
18 model.modelNode.create('comp1');
19
20 model.file.clear;
21
22 model.geom.create('geom1', 2);
23
24 model.mesh.create('mesh1', 'geom1');
25
26 semiax_big = 1;
27 semiax = [semiax_big (2/pi)/semiax_big]; %pi*a*b = 2
28 a = semiax(1);
29 b = semiax(2);
30 model.geom('geom1').create('e1', 'Ellipse');
31 %model.geom('geom1').feature('e1').set('semiaxes', {'1.3' '0.9'});
32 model.geom('geom1').feature('e1').set('semiaxes', {num2str(semiax)});
33 model.geom('geom1').feature('e1').set('pos', {'0' '0'});
34 %substitute it by a loop-----------------------------------------------
35 %model.geom('geom1').create('r1', 'Rectangle');
36 %model.geom('geom1').feature('r1').set('size', {'0.5' '0.3'});
37 %model.geom('geom1').feature('r1').set('pos', {'-0.6' '-0.4'});
38 %model.geom('geom1').create('r2', 'Rectangle');
39 %model.geom('geom1').feature('r2').set('size', {'0.5' '0.3'});
40 %model.geom('geom1').feature('r2').set('pos', {'-0.6' '0.1'});
41 %model.geom('geom1').create('r3', 'Rectangle');
42 %model.geom('geom1').feature('r3').set('size', {'0.5' '0.3'});
43 %model.geom('geom1').feature('r3').set('pos', {'0.1' '-0.4'});
44 %model.geom('geom1').create('r4', 'Rectangle');
45 %model.geom('geom1').feature('r4').set('size', {'0.5' '0.3'});
46 %model.geom('geom1').feature('r4').set('pos', {'0.1' '0.1'});
47 %----------------------------------------------------------------------
48 alpha = 1;
49 lados = eps^alpha * [(1/4)/0.4 0.4];
50
51 nRec = 0;
```

```
sdifference = ...
    'model.geom(''geom1'').feature(''dif1'').selection(''input2'').set({';
for j = 1 : 2*semiax(2)/eps + 1
    %+1 is for cases when the periodicity cell doesn't fit into the
    %ellipse, but the rectangle does

    for i = 1 : 2*semiax(1)/eps + 1
        %+1 is for cases when the periodicity cell doesn't fit into the
        %ellipse, but the rectangle does

        l_down = [-semiax(1) + (i-1)*eps + (eps - lados(1))/2, ...
            -semiax(2) + (j-1)*eps + (eps - lados(2))/2];
        r_down = l_down + [lados(1) 0];
        l_up = l_down + [0 lados(2)];
        r_up = l_down + lados;

        if in_Elipse(l_down, a, b) && in_Elipse(r_down, a, b) && ...
            in_Elipse(l_up, a, b) && in_Elipse(r_up, a, b)
            nRec = nRec + 1;
            nombR = ['r' num2str(nRec)];
            model.geom('geom1').create(nombR, 'Rectangle');
            model.geom('geom1').feature(nombR).set('size', { ...
                num2str(lados) });
            model.geom('geom1').feature(nombR).set('pos', { ...
                num2str(l_down) });
            sdifference = [sdifference ' ''', nombR, ''' '];
        end
    end
end
sdifference = [sdifference, '});'];
model.geom('geom1').create('dif1', 'Difference');
model.geom('geom1').feature('dif1').selection('input').set({'e1'});

%add rectangles---------
eval(sdifference);

model.geom('geom1').create('sca2', 'Scale');
model.geom('geom1').feature('sca2').set('pos', {'-1.2' '0'});
model.geom('geom1').feature('sca2').set('factor', '1.0000001192092896');
model.geom('geom1').feature('sca2').set('isotropic', '1.0000001192092896');
model.geom('geom1').feature('sca2').selection('input').set({'dif1'});
model.geom('geom1').run;

model.physics.create('lpeq', 'LaplaceEquation', 'geom1');
model.physics('lpeq').create('dir1', 'DirichletBoundary', 1);

model.physics('lpeq').feature('dir1').selection.set([4*nRec + 1 : 4 * ...
    (nRec+1)]);
model.physics('lpeq').create('flux1', 'FluxBoundary', 1);
model.physics('lpeq').feature('flux1').selection.set([1:4*nRec]);

model.view('view1').axis.set('abstractviewxscale', '0.007860420271754265');
model.view('view1').axis.set('ymin', '-1.1947839260101318');
model.view('view1').axis.set('xmax', '1.9729712009429932');
model.view('view1').axis.set('abstractviewyscale', '0.007860420271754265');
model.view('view1').axis.set('abstractviewbratio', '-0.16158542037010193');
model.view('view1').axis.set('abstractviewtratio', '0.16158542037010193');
model.view('view1').axis.set('abstractviewrratio', '0.3661609888076782');
model.view('view1').axis.set('xmin', '-1.9729595184326172');
model.view('view1').axis.set('abstractviewlratio', '-0.3661547303199768');
```

```
model.view('view1').axis.set('ymax', '1.1947839260101318');

model.physics('lpeq').feature('dir1').set('r', '1');
model.physics('lpeq').feature('flux1').set('g', [num2str(eps),'*(-u)']);

model.mesh('mesh1').autoMeshSize(2); %extrafine mesh

model.mesh('mesh1').run;

model.study.create('std1');
model.study('std1').create('stat', 'Stationary');

model.sol.create('sol1');
model.sol('sol1').study('std1');
model.sol('sol1').attach('std1');
model.sol('sol1').create('st1', 'StudyStep');
model.sol('sol1').create('v1', 'Variables');
model.sol('sol1').create('s1', 'Stationary');
model.sol('sol1').feature('s1').create('fc1', 'FullyCoupled');
model.sol('sol1').feature('s1').feature.remove('fcDef');

model.study('std1').feature('stat').set('initstudyhide', 'on');
model.study('std1').feature('stat').set('initsolhide', 'on');
model.study('std1').feature('stat').set('solnumhide', 'on');
model.study('std1').feature('stat').set('notstudyhide', 'on');
model.study('std1').feature('stat').set('notsolhide', 'on');
model.study('std1').feature('stat').set('notsolnumhide', 'on');

model.result.create('pg1', 'PlotGroup2D');
model.result('pg1').create('surf1', 'Surface');

model.sol('sol1').attach('std1');
model.sol('sol1').runAll;


model.result.table.create('tbl1', 'Table');

%average----------------------------------------
model.result.numerical.create('av1', 'AvLine');
model.result.numerical('av1').selection.set([1:4*nRec]);
model.result.numerical('av1').set('probetag', 'none');
model.result.numerical('av1').set('expr', 'u');
model.result.numerical('av1').set('unit', '');
model.result.numerical('av1').set('method', 'integration');
model.result.numerical('av1').set('table', 'tbl1');
model.result.numerical('av1').setResult
%-----------------------------------------------


out1 = model;

thetable = mphtable(model, 'tbl1');
out2 = thetable.data;

end


function Inside = in_Elipse(P, a, b) %centered in (0,0)
% Check the point

```

```
166     Inside = ( (P(1)/a)^2 + (P(2)/b)^2 ) < 1;
167
168 end
```

D Code for the comparison

```
1  clear all
2
3  % Homogenized solution
4
5  modelh = Ellipsis_andRect_converg;
6
7  datah = mpheval(modelh, 'u');
8
9  ph = datah.p;
10 Xh = ph(1,:);
11 Yh = ph(2,:);
12 Uh = datah.d1;
13
14 semiax_big = 1;
15 semiax = [semiax_big (2/pi)/semiax_big]; %pi*a*b = 2
16 a = semiax(1);
17 b = semiax(2);
18 sz_mesh = 0.01;
19 [Xqh, Yqh] = meshgrid(-a:sz_mesh:a, -b:sz_mesh:b);
20 %force the mesh we had to fit coordinates, for plotting it
21
22 Vqh = griddata(Xh, Yh, Uh, Xqh, Yqh);
23 %surf(Xqh, Yqh, Vqh)
24
25 Vqh (isnan(Vqh)) = 1;
26 integh = sum(sum(Vqh)) * sz_mesh^2;
27
28 %--------------------------------------------------------------------
29
30 % we resolve for different epsilon and compare with the homogenized ...
       solution
31 % eps = 0.2;
32
33 eps_v = [1/3 1/5 1/7 1/12 1/18];
34
35 figure(1)
36 subplot(2, 3, length(eps_v)+1) %I plot it before the loop to avoid to ...
       overwrite pg1
37 mphplot(modelh,'pg1')
38 title('Homogenized problem')
39
40 for n = 1:length(eps_v)
41     [model eta(n)] = Ellipsis_andRect_addAve(eps_v(n));
42     %plots-----------------------------------------
43     subplot(2,3,n)
44     mphplot(model,'pg1')
45     title(['\epsilon = 1/' num2str(1/eps_v(n))])
46     %----------------------------------------------
47
48     data = mpheval(model, 'u');
```

```
49      p = data.p;
50      X = p(1,:);
51      Y = p(2,:);
52      U = data.d1;
53
54      [Xq, Yq] = meshgrid(-a:sz_mesh:a, -b:sz_mesh:b);
55      %force the mesh we had to fit coordinates, for plotting it
56      Vq = griddata(X, Y, U, Xq, Yq);
57      %surf(Xq, Yq, Vq)
58
59      Vq (isnan(Vq)) = 1;
60      integL2 = sum(sum((Vq-Vqh).^2)) * sz_mesh^2;
61      convergL2(n) = integL2;
62      integLinf = max(max(abs(Vq-Vqh)));
63      convergLinf(n) = integLinf;
64  end
65
66  figure(2)
67  plot(1./eps_v, eta)
68  xlabel('1/\epsilon')
69  ylabel('\eta_{\epsilon}')
70  title('convergence efficiency')
71  hold on
72  ceta = eps_v*0 + 0.79684;
73  plot(1./eps_v, ceta,'r')
74  axis([3 18 0.795 0.82])
75  legend('\eta_{\epsilon}', '\eta')
76
77  figure(3)
78
79  % L2 norm
80  subplot(1,2,1)
81  plot(1./eps_v, convergL2)
82  title('convergence solution L_2')
83  xlabel('1/\epsilon')
84  ylabel('{||u_{\epsilon}-u||}_{2}^2')
85
86  % L infinite norm
87  subplot(1,2,2)
88  plot(1./eps_v, convergL2)
89  title('convergence solution L_{\infty}')
90  xlabel('1/\epsilon')
91  ylabel('{||u_{\epsilon}-u||}_{\infty}')
```