UCL Université catholique de Louvain

École polytechnique de Louvain (EPL)

epl

Title of the master thesis

Subtitle (optional)

Dissertation presented by
Firstname LASTNAME , Firstname LASTNAME

for obtaining the Master's degree in
Speciality

Supervisor(s)
Firstname LASTNAME, Firstname LASTNAME

Reader(s)
Firstname LASTNAME, Firstname LASTNAME , Firstname LASTNAME

Academic year 20..-20..

Chapter 1

Results and discussion

In this chapter, we will present the different results and discuss them.

Expliquer ce qu'on va faire et tester

Mettre ici les trois "super mesh"

1.1 Multigrid

In this section, we will test the coarse part of the preconditioner : the multigrid solver. This will be done in two steps. First, we will verify a well known property of the multigrid solvers : the h-independent convergence. We will also compare the number and iterations needed while varying key parameters of the model. Those tests will be performed on various meshes. The second part will focus on the influence of hanging nodes on the numerical solution.

Let us before all present a type of numerical solution that can be obtained using the multigrid solver. Figure 1.1 shows an example of the numerical solution computed. We can see that even with $p = 1$, we have a good approximation. This is because the forcing term is not at all oscillatory.

1.1.1 H-independent convergence

Let us first verify that our geometric multigrid solver has the required property and that the same number of iterations is needed to obtain a given accuracy, however small the elements. We will use the model problem throughout this section with the same right hand side. For all the tests below, the domain will be : $\Omega = [-1; 1]^2$. We will solve :

$$\nabla^2 u = -\frac{\pi^2}{2}\cos(\frac{\pi}{2}x)\cos(\frac{\pi}{2}y) \qquad \text{on } \Omega \tag{1.1}$$

$$u = 0 \qquad \text{on } \Gamma \tag{1.2}$$

It is easy to see that for the given domain, we have an analytic solution :

$$u(x, y) = \cos(\frac{\pi}{2}x)\cos(\frac{\pi}{2}y))$$

Let us now explain how we define the error. We will look at the absolute difference between the value of the approximation and the value of the analytic solution at the global nodes and take the maximum. Formally, we have that the error after iteration k, e_k is :

$$e_k = \max_i |u(x_i, y_i) - u_i^k|$$

Where u_i^k is the value of our approximation at the global node i after iteration k. Since $u_i^0 = 0$ for all i, it is clear that $e_0 = 1$.

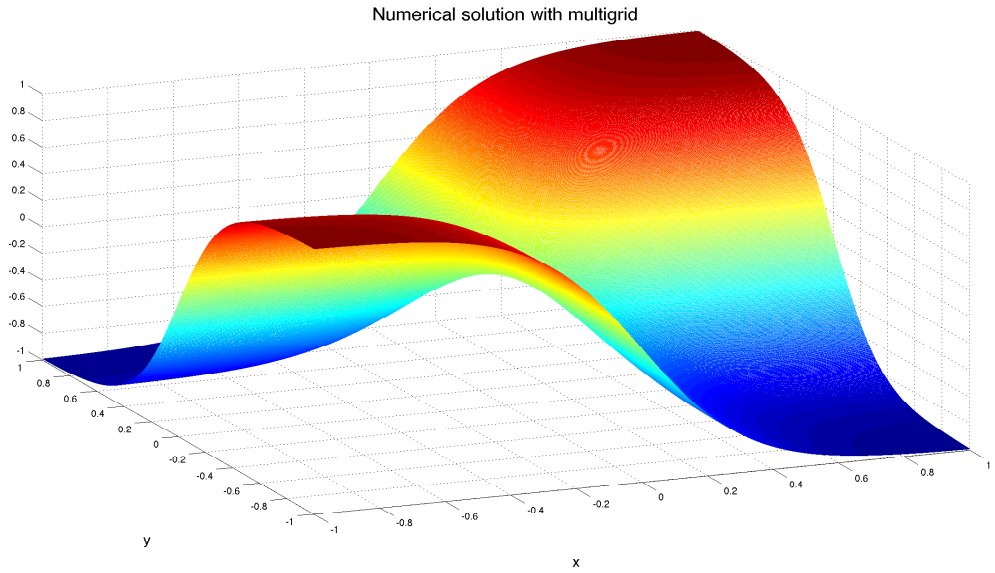


Figure 1.1: Numerical solution using the multigrid solver of $\nabla^2 u = f$ for $f = -2\tanh(3x)\tanh(3y)(18 - 9\tanh^2(3x) - 9\tanh^2(3y))$

Figure 1.2 shows the two "supra meshes" that we will refine during the tests. Some refinements will be uniform and some will be so that we have the presence of hanging nodes. We can note that even for the crooked mesh, the elements are not really distorted. It does not matter since we are only testing the h-independent convergence. Having a mesh with elements that are more distorted will only influence the accuracy of the approximation and not how the algorithm solve the linear system we want to solve.

Let us start with a simple V-cycle on the regular mesh (figure 1.2a) that we will refine uniformly. We will compare the errors when we increase the number of degrees of freedom and for different ν_1 and ν_2. The sum of the two smoothing parameters is chosen to be constant so that we have the same number of Jacobi iterations for all pairs ν_1 and ν_2. Here, we chose the sum to be equal to four.

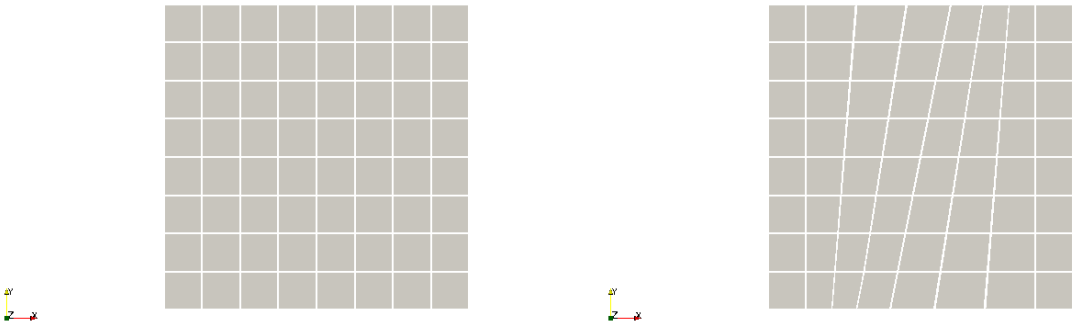
The results are shown in table 1.1. We can see that we indeed have an h-independent convergence of the solver. For every pair of the smoothing parameters, at least for the first few iterations, the error e_k is identical for all values of N. Except for the pair $\nu_1 = 4$ and $\nu_2 = 0$, each iteration roughly decrease the error by one decimal point.

We can also note that the values of the parameters influence the convergence. For this particular problem and this particular mesh, the values $\nu_1 = 0$ and $\nu_2 = 4$ seem to be the best as the error is smaller after the same number of iterations than for the other pairs. The less post smoothing iterations (ν_2) we do, the slower the convergence. This is true for $\nu_2 = 1$ where the error on the finest mesh after six iterations is a hundred times larger than on the same mesh after the same number of iterations for $\nu_2 = 4$, and it is clearer still for $\nu_2 = 0$ where the error is a thousand times larger after six iterations on the finest mesh.

We have to note that even tough the errors are identical for all meshes at first, we have a difference after a few iterations. This can be explained by the fact that our geometric multigrid algorithm actually solves a linear system whereas the error is measured as the difference between the analytic solution and the solution of the linear system. Thus, even if we solved the linear system exactly, we would still have an error and that error should decrease as the number of degrees of freedom increases. This is indeed what we observe here. For example, for the mesh with $N = 2.6\,10^5$, we can see that after six iterations, we have almost converged and that the error stays around $3.14\,10^{-6}$. Even if we did several more iterations, the error would not decrease significantly. That is because we have the solution of the linear system and the error is only due

N	$2.6\ 10^5$	$1.1\ 10^6$	$4.2\ 10^6$	$1.7\ 10^7$	$6.7\ 10^7$
	$\nu_1 = 2$	$\nu_2 = 2$			
e_1	3.56e-02	3.56e-02	3.56e-02	3.56e-02	3.56e-02
e_2	1.34e-03	1.34e-03	1.34e-03	1.34e-03	1.34e-03
e_3	5.42e-05	5.66e-05	5.72e-05	5.73e-05	5.73e-05
e_4	3.11e-06	2.90e-06	3.45e-06	3.59e-06	3.62e-06
e_5	3.30e-06	9.48e-07	3.62e-07	3.50e-07	3.85e-07
e_6	3.17e-06	8.14e-07	2.26e-07	8.26e-08	5.23e-08
	$\nu_1 = 3$	$\nu_2 = 1$			
e_1	3.70e-02	3.71e-02	3.71e-02	3.71e-02	3.71e-02
e_2	1.62e-03	1.63e-03	1.63e-03	1.63e-03	1.63e-03
e_3	1.04e-04	1.06e-04	1.07e-04	1.07e-04	1.07e-04
e_4	1.10e-05	1.22e-05	1.27e-05	1.29e-05	1.29e-05
e_5	4.47e-06	2.20e-06	1.96e-06	2.10e-06	2.13e-06
e_6	3.34e-06	1.00e-06	4.38e-07	3.53e-07	3.88e-07
	$\nu_1 = 1$	$\nu_2 = 3$			
e_1	3.57e-02	3.57e-02	3.57e-02	3.57e-02	3.57e-02
e_2	1.29e-03	1.29e-03	1.29e-03	1.29e-03	1.29e-03
e_3	4.66e-05	4.89e-05	4.95e-05	4.96e-05	4.97e-05
e_4	1.53e-06	1.53e-06	2.10e-06	2.24e-06	2.28e-06
e_5	3.08e-06	7.31e-07	1.43e-07	1.17e-07	1.51e-07
e_6	3.14e-06	7.83e-07	1.95e-07	4.80e-08	1.13e-08
	$\nu_1 = 4$	$\nu_2 = 0$			
e_1	4.55e-02	4.55e-02	4.55e-02	4.55e-02	4.55e-02
e_2	3.26e-03	3.26e-03	3.26e-03	3.26e-03	3.26e-03
e_3	4.64e-04	4.71e-04	4.71e-04	4.71e-04	4.71e-04
e_4	9.24e-05	9.60e-05	9.65e-05	9.67e-05	9.67e-05
e_5	1.74e-05	2.04e-05	2.12e-05	2.14e-05	2.14e-05
e_6	6.17e-06	3.86e-06	4.55e-06	4.74e-06	4.78e-06
	$\nu_1 = 0$	$\nu_2 = 4$			
e_1	3.58e-02	3.58e-02	3.58e-02	3.58e-02	3.58e-02
e_2	1.29e-03	1.29e-03	1.29e-03	1.29e-03	1.29e-03
e_3	4.53e-05	4.77e-05	4.82e-05	4.84e-05	4.84e-05
e_4	1.17e-06	1.31e-06	1.89e-06	2.03e-06	2.07e-06
e_5	3.03e-06	6.80e-07	9.14e-08	7.66e-08	1.12e-07
e_6	3.13e-06	7.76e-07	1.87e-07	4.04e-08	3.62e-09

Table 1.1: Errors after k iterations of a V-cycle (e_k) for the regular mesh uniformly refined to have N degrees of freedom and for different values of the parameters ν_1 and ν_2



(a) Regular mesh

(b) Crooked mesh

Figure 1.2: The two "supra meshes" that will be refined during the tests for the multigrid solver. We have one regular mesh (left) where the elements are squares and one crooked mesh (right) where the elements are slightly distorted.

N	$2.6\,10^5$	$1.1\,10^6$	$4.2\,10^6$	$1.7\,10^7$	$6.7\,10^7$
	$\nu_1 = 2$	$\nu_2 = 2$			
e_1	3.64e-02	3.64e-02	3.64e-02	3.64e-02	3.64e-02
e_2	2.56e-03	2.49e-03	2.46e-03	2.44e-03	2.43e-03
e_3	7.41e-04	6.33e-04	5.80e-04	5.54e-04	5.41e-04
e_4	6.15e-04	3.15e-04	1.62e-04	1.35e-04	1.28e-04
e_5	5.96e-04	3.01e-04	1.52e-04	7.67e-05	4.76e-05
e_6	5.91e-04	2.98e-04	1.50e-04	7.50e-05	3.77e-05

Table 1.2: Errors after k iterations of a V-cycle (e_k) for the crooked mesh uniformly refined to have N degrees of freedom and for $\nu_1 = 2$ and $\nu_2 = 2$

to the discretization. If we refine the mesh and go to $N = 6.7\,10^7$, then we can get smaller errors (of the order of 10^{-8}).

Let us now explore the results for the crooked mesh (figure 1.2b). Here also, we should expect an h-independent convergence. We only show the results for $\nu_1 = 2$ and $\nu_2 = 2$ but the same commentary applies for the other pairs. The results can be seen on table 1.2

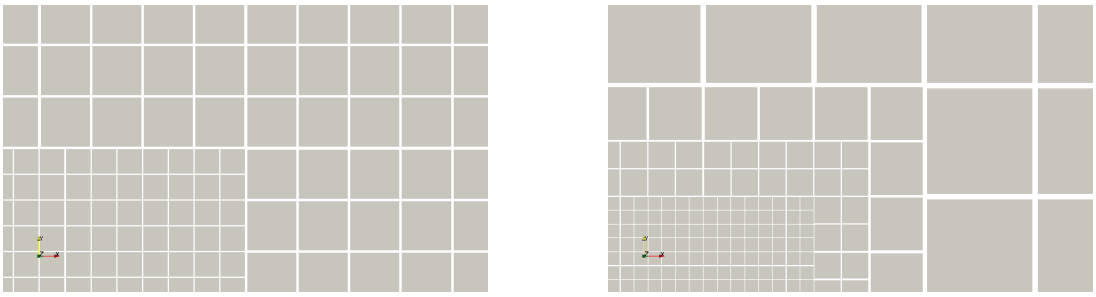
We can see that the for the first few iterations, the error e_k is independent of the mesh. However, the note we made earlier is much clearer here. Because the mesh is not regular, the effect of discretization are more important and therefore we will not reach the same accuracy than we did before. That is why after six iterations, the less refined grid ($N = 2.6\,10^5$) still has an error of $5.91\,10^{-4}$. More iterations will not have a great impact on the solution since the error is mostly due to the discretization.

1.1.2 Influence of hanging nodes

We will now investigate the influence of hanging nodes on our solution. We will present the results only for the regular mesh but the tests have been performed on both and the same conclusions apply to the crooked mesh.

We will compare the values of the error between one mesh with no hanging nodes, one where we only have refined the lower left part of the domain once, and one where we have a rapid transition between two parts of different refinement level (which will occur often in AMR).

Figure 1.3 presents the latter two meshes. For the mesh in figure 1.3b, we have refined thrice more in a certain region than in the adjacent one. Since we do not allow adjacent quadrants to



(a) Two levels

(b) Levels rapidly changing

Figure 1.3: Zoom on a certain part of the two meshes containing hanging nodes that will be used to test the multigrid solver. We have one mesh where we only have refined a part of the domain once more than the rest (left) and a mesh where we have refined a part thrice more than the rest which results in levels changing rapidly (right).

Meshes	No hanging nodes	Figure 1.3a	Figure 1.3b
N	$4.2\ 10^6$	$7.3\ 10^6$	$7.0\ 10^7$
e_1	3.56e-02	3.56e-02	3.56e-02
e_2	1.34e-03	1.34e-03	1.34e-03
e_3	5.72e-05	5.72e-05	5.74e-05
e_4	3.45e-06	3.47e-06	3.64e-06

Table 1.3: Errors after k iterations of a V-cycle (e_k) for a mesh without hanging nodes and the two meshes presented in figure 1.3. Each mesh has N degrees of freedom and the smoothing parameters were $\nu_1 = 2$ and $\nu_2 = 2$.

be more than one level apart, we obtain a "layer" where the levels of the quadrants is rapidly changing. In order to compare solutions, we made sure that the largest quadrants in all three meshes had the same size. This means that the meshes with hanging nodes have a lot more degrees of freedom.

Table 1.3 shows the results for the three different meshes using a V-cycle and with the smoothing parameters $\nu_1 = 2$ and $\nu_2 = 2$. Here again, we can see that the convergence is h-independent and that one iteration gives us roughly one more decimal. The presence of hanging nodes has no influence on the error we observe after a given number of iterations. The same result is observed with all pairs of the smoothing parameters and with other meshes.

This will be important in the next sections when we will use our multigrid solver as a preconditioner. Indeed, we will see that it is the coarse correction that allow for h-independent convergence.

1.2 Fine preconditioner

Let us now move on to the fine part of the preconditioner : the overlapping additive Schwarz preconditioner. We will test it by using the preconditioned conjugate gradients method described earlier but, for now, the preconditioner will only consist of the fine part (i.e. $P = P^f$).

As in the previous section, we will perform the tests in two parts : first, we will use meshes with elements that are distorted or not but with no hanging nodes. Then, we will see how the fine preconditioner performs in the presence of hanging nodes also for meshes that are distorted or not. Here, of course, we will use interpolations of higher degree. Typically, the tests will be performed for $p = 2, 4, 6, 8$.

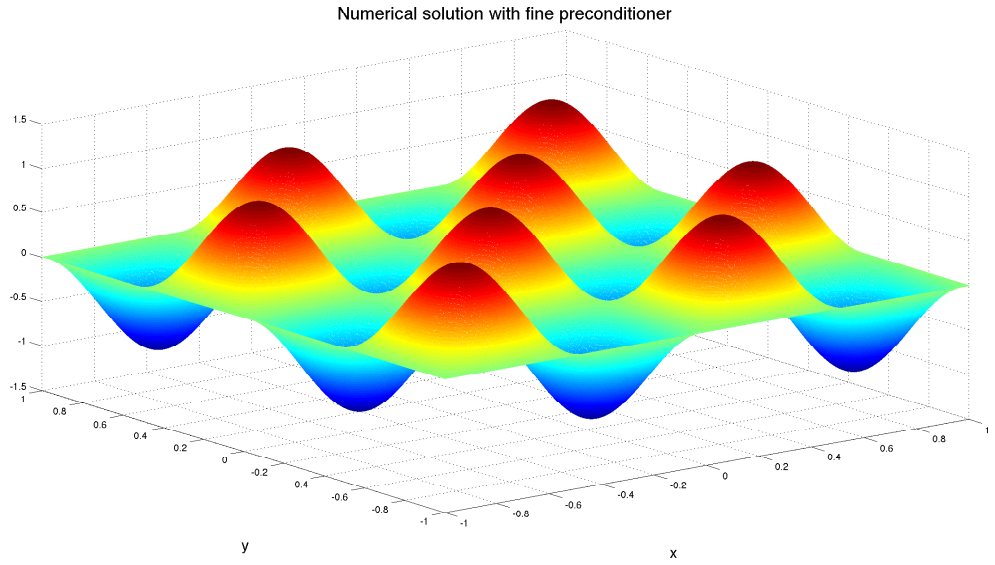


Figure 1.4: Numerical solution to problem 1.3 using an interpolation of order $p = 2$ and $1.0\,10^6$ degrees of freedom on a regular mesh with no hanging nodes.

1.2.1 No hanging nodes

Let us first present the problem we will use throughout this section. The forcing term will be chosen more oscillatory than in the previous part since we use interpolations of higher degree. As before, the domain is : $\Omega = [-1; 1]^2$ and Γ is the boundary. The problem is :

$$\nabla^2 u = -8\pi^2 \sin(2\pi x) \sin(2\pi y) \qquad \text{on } \Omega \tag{1.3}$$

$$u = 0 \qquad \text{on } \Gamma \tag{1.4}$$

This problem has an analytic solution and it is easy to convince oneself that this solution is given by :

$$u(x, y) = \sin(2\pi x) \sin(2\pi y)$$

Figure 1.4 shows the numerical solution to the problem above for $p = 2$ and $1.0\,10^6$ degrees of freedom for a regular mesh. We can note that it is exactly the same number of degrees of freedom as if we had refined uniformly once more and used an interpolation of degree $p = 1$. Let us then compare how the two approximations perform. We solved the problem for $p = 1$ with our multigrid solver and the problem for $p = 2$ with the PCG and the fine preconditioner. Let us denote u_i^j as the value of the approximation for $p = j$ at node i. We have that :

$$e^1 = \max_i |u_i^1 - u(x_i, y_i)| = 5.02\,10^{-5}$$

$$e^2 = \max_i |u_i^2 - u(x_i, y_i)| = 1.01\,10^{-9}$$

We can see that with the same number of degrees of freedom, an approximation using $p = 2$ is much more accurate. This is because the solution is really smooth and is better approximated using a higher order interpolation than a bilinear interpolation on smaller quadrants. This is one example of the reasons we want to use higher order interpolations.

Number of quadrants	16^2	32^2	64^2	128^2	256^2
$p = 2$	$1.1\,10^3$	$4.2\,10^3$	$1.7\,10^4$	$6.6\,10^4$	$2.6\,10^5$
$p = 4$	$4.2\,10^3$	$1.7\,10^4$	$6.6\,10^4$	$2.6\,10^5$	$1.1\,10^6$
$p = 6$	$9.4\,10^3$	$3.7\,10^4$	$1.5\,10^5$	$5.9\,10^5$	$2.4\,10^6$
$p = 8$	$1.7\,10^4$	$6.6\,10^4$	$2.6\,10^5$	$1.1\,10^6$	$4.2\,10^6$

Table 1.4: Number of degrees of freedom for a regular mesh with different number of quadrants and for different degrees of interpolation.

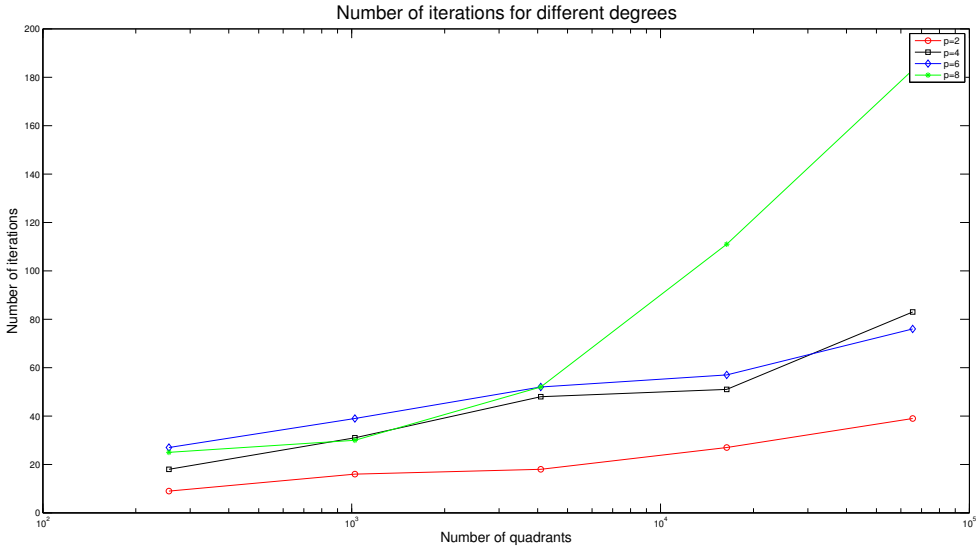


Figure 1.5: Number of iterations of PCG with only the fine preconditioner for different degree p of interpolation as a function of the number of quadrants in a regular mesh.

Regular meshes

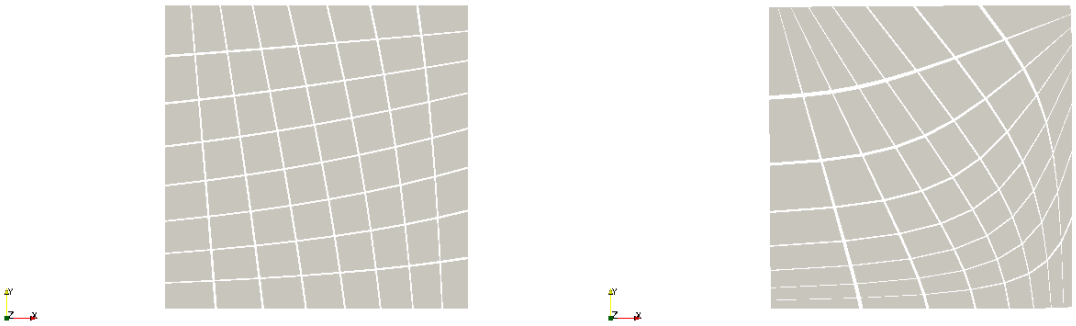
Let us now move on to the comparison for different degrees of the number of iteration needed to reach a given accuracy as a function of the number of quadrants. We will take our regular mesh and uniformly refine it. Then, for $p = 2, 4, 6, 8$, we will see how many iterations are needed to reach a given error on the norm of the residual. Let us denote r_k the residual after iteration k of the preconditioned conjugate gradients. Of course, since our initial guess is zero, we have that $r_0 = b$ (since we are solving the linear system $Au = b$). For the following tests, our stopping criterion is given by :

$$\frac{||r_k||_2}{||r_0||_2} < 10^{-3}$$

Figure 1.5 shows the results. To put the data in perspective, we also have to show the number of degrees of freedom. Indeed, for a given number of quadrants, the higher degree the interpolation is, the more nodes we have. Table 1.4 contains the number of nodes for each mesh and for each degree p.

We can see that, even without the coarse preconditioner, we are solving the system in a small number of iterations compared to the number of degrees of freedom. For example, we only do about 80 iterations to solve the system with $2.4\,10^6$ degrees of freedom and with interpolations of degree $p = 6$.

We can also see that for every degree, the number of iterations increases when we refine the mesh. This is to be expected since the information from the boundaries has to go through more quadrant before propagate to the entire domain. Asymptotically, the number of iterations is



(a) Mesh with $a = 1.1$

(b) Mesh with $a = 1.4$

Figure 1.6: Examples of the meshes used for the tests of the fine preconditioner with distorted elements. The meshes are created using GMSH with its progression tool. The key parameter is the common ratio a that we will be increasing progressively.

expected to double as the number of quadrants is multiplied by four (i.e. the mesh size is divided by two). We can see that it is not yet the case here.

A last remark we can make is that the number of iterations tends to increase when the degree of the interpolation increases. This is especially true for the finest mesh where we need 183 iterations for $p = 8$ where we only need 39 iterations for $p = 2$. This can be explained by the fact that the size of the overlap decreases when p grows. As mentioned in Ref here!!, this issue would be fixed if we imposed a constant overlap.

Meshes with distorted elements

Let us now move on to meshes that are not regular anymore. Let us remember that when we developed the fine preconditioner, we assumed that the elements were rectangular which allowed us to compute the analytic solution to the problem. This part explores the influence of having distorted elements on the number of iterations needed to obtain a given accuracy.

To control the deformation, we will continually deform the regular mesh using the progression tool of GMSH ref here!!. The deformation is performed using a geometric progression, using the common ratio a as the parameter. Obviously, the higher the parameter a, the more distorted the mesh is. It is clear that for $a = 1$, we have a regular mesh. Figure 1.6 presents three meshes of obtained with the progression with GMSH. On the left we used $a = 1.1$ and on the right we used $a = 1.4$.

Using the same tolerance as in the previous part, we ran the preconditioned conjugate gradients with the fine preconditioner for those new meshes. The order of the interpolation used is $p = 2$. The results are given in figure 1.7.

We can see, as it was expected, that the more we deform the mesh, the more iterations we need to do in order to obtain the wanted accuracy. The fact that, for distorted elements, we do not invert exactly the system but use an approximation allows us to compute the fine preconditioner efficiently even when we have a lot of elements but we can see here that it also costs more iterations of the preconditioned conjugate gradients. However, the gain is still huge. Indeed, for a degree of interpolation p, we have $(p+3)^2$ nodes per overlapping subdomain. This means a complexity of $\mathcal{O}((p+3)^6)$ to solve the system exactly. Instead, with the method we use, the only need to do a few matrix multiplications where the matrices have $(p+3)$ rows and columns. This means a complexity of $\mathcal{O}((p+3)^2)$. Even with $p = 2$, the results shows that it is much faster to not take geometric factors into account.

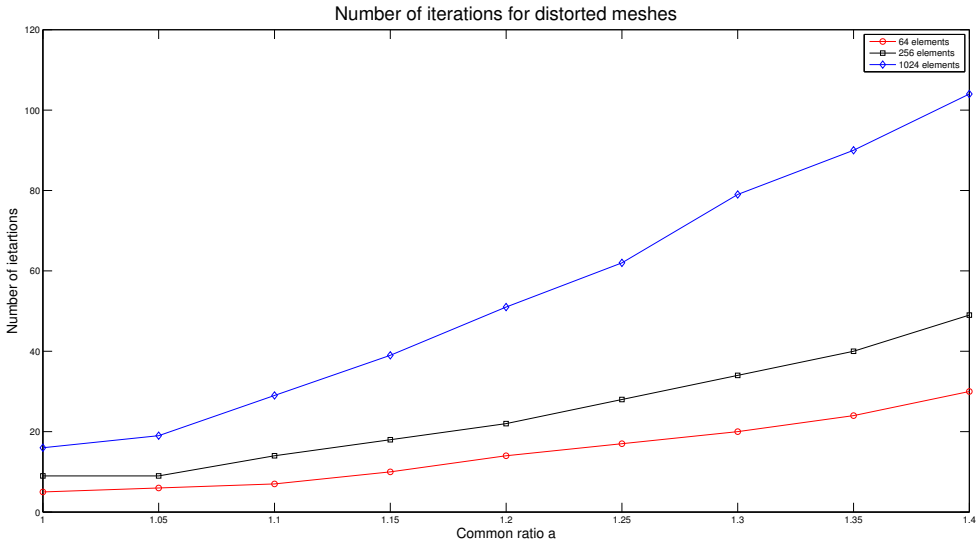


Figure 1.7: bla

We can also note that the effect of increasing the number of elements (i.e. reducing the mesh size) on the number of iterations is clearer here. For example, for the mesh with $a = 1.15$, we need 10 iterations for 8^2 elements, 18 iterations for 16^2 elements and 39 iterations for 32^2 elements. The same explanation applies here : when we multiply the number of elements by four, we roughly multiply the number of quadrants in each direction by two and therefore the information needs twice as many iterations to propagate to the domain.

1.2.2 Influence of hanging nodes

In this part, we will explore the influence of hanging nodes on the number of iterations needed by the preconditioned conjugate gradients with the fine preconditioner to converge. Because we want meshes that are not artificial and come from real AMR applications, we will change the forcing term in this part and we will also use a recursive refine function when we build the mesh with p4est.

Let us first define the problem and the forcing term. As before, the domain is : $\Omega = [-1; 1]^2$ and Γ is the boundary. We will solve :

$$\nabla^2 u = -2\tanh(nx)\tanh(my)\left[n^2(1-\tanh(nx)^2) + m^2(1-\tanh(my)^2)\right] \quad \text{on } \Omega \qquad (1.5)$$

$$u = \tanh(nx)\tanh(my) \quad \text{on } \Gamma \qquad (1.6)$$

We can see that problem 1.5 has an analytic solution that is given by :

$$u(x, y) = \tanh(nx)\tanh(my)$$

The parameters n and m can be adjusted to make the jump in the hyperbolic tangent more steep. An example of a numerical solution with $p = 1$ and obtained by our multigrid solver has already been shown on figure 1.1 for $n = 3$ and $m = 3$.

As explained before, we will use a recursive refine function when we build the forest. Since we know the analytic solution, we can cheat a little and use it for the refinement process. We will ask that the absolute value of the difference between the value of u in the center of the quadrant and the mean of the values of u at the four corners of the quadrant is less than a fixed tolerance

Tol	0.020	0.015	0.010
Number of quadrants	1276	1384	3064
hang	14.05%	16.61%	22.13%

Table 1.5: Number of quadrants obtained using the recursive refine function on the problem 1.5 for different tolerances as well as the ratio *hang* for each mesh with an interpolation of degree $p = 2$.

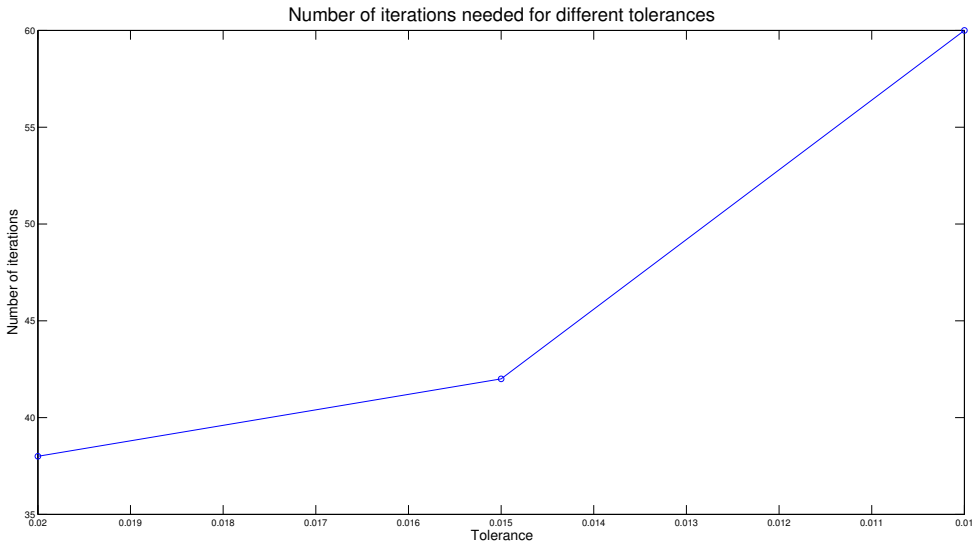


Figure 1.8: Number of iterations needed to reach a given norm on the residual for a degree of interpolation $p = 2$ and for meshes obtained with different tolerances.

multiplied by the maximum value of the function. So, if the four corners have coordinates (x_i, y_i) for $i = 0, 1, 2, 3$, and that $u_{max} = max_{x,y\in\Omega} |u|$, we impose the following rule :

$$\left| u(\frac{1}{4}\sum_{i=0}^{3} x_i, \frac{1}{4}\sum_{i=0}^{3} y_i) - \frac{1}{4}\sum_{i=0}^{3} u(x_i, y_i) < u_{max} tol \right|$$

Where *tol* is a fixed tolerance. Intuitively, the tighter the tolerance, the more refined the grid needs to be and the more hanging nodes we will have thanks to the jump in the hyperbolic tangent function.

Increasing the relative number of hanging nodes

To have a rather steep jump, we chose $n = m = 12$ for the different tests that follow. We varied the tolerance to have more or less hanging nodes. We also needed a way to quantify the presence of hanging nodes. Let us define *hang*, the ratio of the number of hanging nodes over the number of global nodes.

$$hang = \frac{\#\text{hanging nodes}}{\#\text{global nodes}}$$

Table 1.5 shows this number for different values of the tolerance and for $p = 2$. We can see, as expected, that the ratio *hang* increases while we make the tolerance tighter. Of course, the exact value of *hang* does not matter since even for a given mesh, it will change with the degree of the interpolation so it is rather how it evolves that interests us.

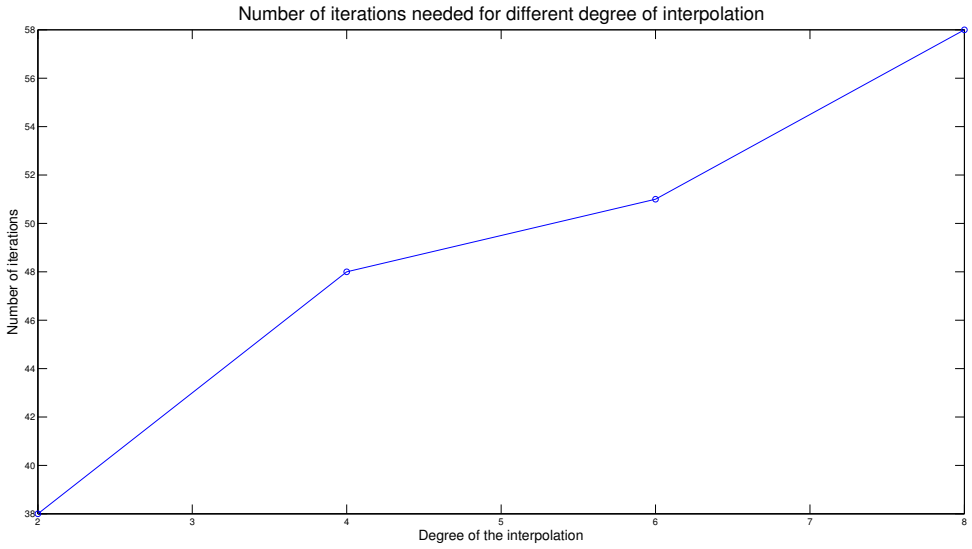


Figure 1.9: Number of iterations needed to reach the tolerance of the norm of the residual as a function of the degree of the interpolation used for a mesh obtained using $tol = 0.02$ in the recursive refine function.

Let us now see how the number of iterations varies for the different meshes. Figure 1.8 shows the number of iterations needed to reach the same norm on the residual as before for the three different meshes obtained with the tolerances presented in table 1.5. We can see on the graph that the number of iterations increases when we have relatively more hanging nodes. We have to be careful since this phenomenon can also be explained by the fact that we have more quadrants when the tolerance gets tighter and that also causes an increase in the number of iterations as showed earlier in this section. However, it is observed that we need less iterations when we do not have hanging nodes and for a similar number of quadrants (for example, for 1024 quadrants without hanging nodes, we need 18 iterations whereas we need 38 for 1276 quadrants).

As explained in the theory chapter, when we have hanging nodes, we do not treat all hanging possibilities and therefore we have an error when we compute the local residual. This explains why we need more iterations in presence of hanging nodes and the fact that we converge less quickly the more hanging nodes we have in the mesh. However, the number of iterations is still acceptable.

Increasing the degree of the interpolation

Let us finally look at what happens when we increase the degree of the interpolation. We tried different degrees of interpolation ($p = 2, 4, 6, 8$) and look at the number of iterations needed to obtain the numerical solution. Figure 1.9 shows the results.

We can see that the number of iterations increases with the degree. We need 38 iterations for $p = 2$ but 58 for $p = 8$. As already explained in the case with no hanging nodes, this is in part due to the fact that when we increase the degree of the interpolation, the size of the overlap decreases and therefore we need more iterations.

We can also mention the fact that since we do not treat all hanging possibilities when we compute the local residual at the overlaps, increasing the degree (and therefore the number of nodes in the overlaps) might have a effect on the number of iterations.

All the tests presented here were on meshes where the elements were not distorted and where we expect the fine preconditioner to behave optimally but the same tests have been performed with distorted quadrants and we observe the same qualitative results.

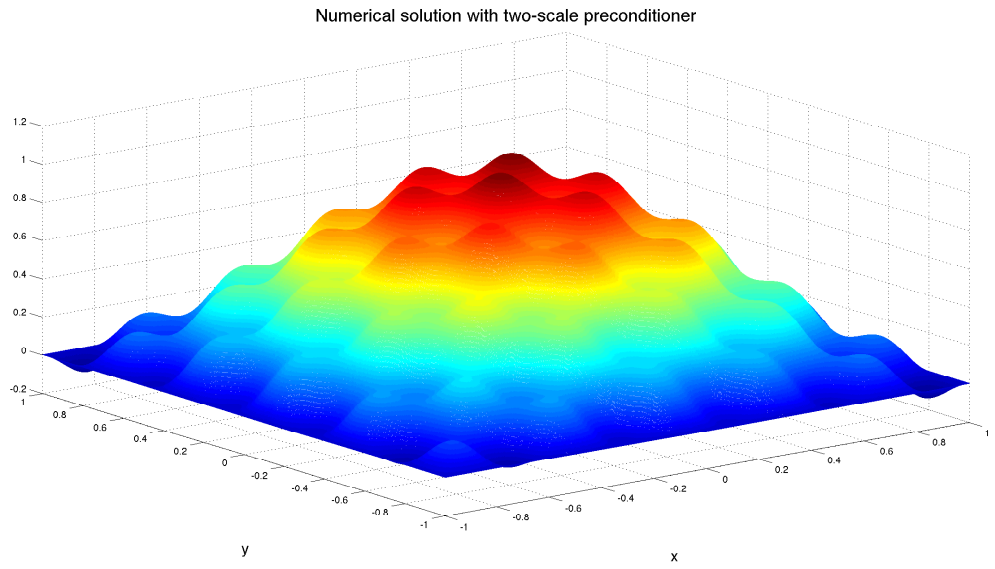


Figure 1.10: Numerical solution to problem 1.7 using an interpolation of order $p = 2$ and $1.0\,10^6$ degrees of freedom on a regular mesh with no hanging nodes. This numerical solution has been obtained by the PCG with the two scale preconditioner.

1.3 Two scale preconditioner

Let us finally move on to the two-scale preconditioner. We showed in the previous part that using only the fine scale preconditioner was not a good idea when the number of quadrants increased (the number of iterations roughly doubles when the mesh size is divided by two). As explained in the theory chapter, the idea is to add a coarse part in the preconditioner (i.e. $P = P^f + P^c$), consisting mainly of solving a problem with an interpolation degree $p = 1$ with a geometric multigrid method, so that the number of iterations stays constant when we increase the number of quandrants.

As in the previous section, the tests will be performed in two parts : first, we will analyse the performances of the two-scale preconditioner when there are no hanging nodes (but for both regular and distorted meshes) and then we will look at what happens when the forest of quadtrees is refined recursively and therefore the mesh is not conforming anymore.

Ajouter si on fait le truc avec les erreurs et le temps

1.3.1 No hanging nodes

Let us first present the problem we will solve in this part. We want our numerical solution to capture both low and high frequency modes so we will superpose a cosine with low frequency and a sine with a high frequency. As before, the domain is : $\Omega = [-1; 1]^2$ and Γ is the boundary. We will solve :

$$\nabla^2 u = -\frac{\pi^2}{2}\cos(\frac{\pi}{2}x)\cos(\frac{\pi}{2}y) - 5\pi^2\sin(5\pi x)\sin(5\pi y) \qquad \text{on } \Omega \qquad (1.7)$$

$$u = 0 \qquad \text{on } \Gamma \qquad (1.8)$$

This problem has an analytic solution that is given by :

$$u(x, y) = \cos(\frac{\pi}{2}x)\cos(\frac{\pi}{2}y) + \frac{1}{10}\sin(5\pi x)\sin(5\pi y)$$

Number of quadrants	128^2	256^2	512^2	1024^2
$p = 2$	$6.6\ 10^4$	$2.6\ 10^5$	$1.1\ 10^6$	$4.2\ 10^6$
$p = 4$	$2.6\ 10^5$	$1.1\ 10^6$	$4.2\ 10^6$	$1.7\ 10^7$
$p = 6$	$5.9\ 10^5$	$2.4\ 10^6$	$9.4\ 10^6$	$3.8\ 10^7$
$p = 8$	$1.1\ 10^6$	$4.2\ 10^6$	$1.7\ 10^7$	$6.7\ 10^7$

Table 1.6: Number of degrees of freedom for a regular mesh with different number of quadrants and for different degrees of interpolation.

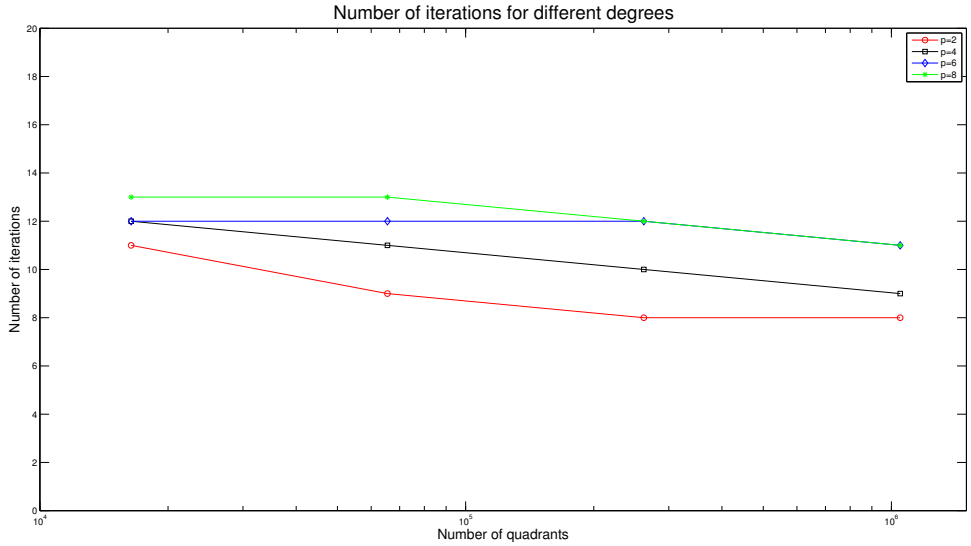


Figure 1.11: Number of iterations of PCG with the two scale preconditioner for degree p of interpolation needed to reach the given tolerance of the norm of the residual as a function of the number of quadrants in a regular mesh.

An example of a numerical solution obtained using the preconditioned conjugate gradients with the two scale preconditioner can be seen on figure 1.10. This solution has been obtained in only 8 iterations and with an interpolation of degree $p = 2$.

Regular meshes

Let us now look at what happens to the number of iterations when we decrease the mesh size for different degrees of interpolation. Table 1.6 shows the number of degrees of freedom for the different meshes used (indeed, for a given mesh, the higher the degree of the interpolation, the more global nodes we have). We can already see that, thanks to the coarse preconditioner, we are able to have a lot more degrees of freedom than in the case where we only had the fine preconditioner.

For the following tests, we also have tighten the tolerance on the norm of the residual. We now require that :

$$\frac{||r_k||_2}{||r_0||_2} < 10^{-5}$$

Figure 1.11 shows the number of iterations needed to reach that tolerance of the norm of the residual when we use our two scale preconditioner for different degrees of interpolation. The first remark we can make is that the number of iterations does not increase with the number of quandrants (as it was the case when we only had the fine preconditioner). So we can conclude that the coarse preconditioner does the job it was designed to do. We can even note that the

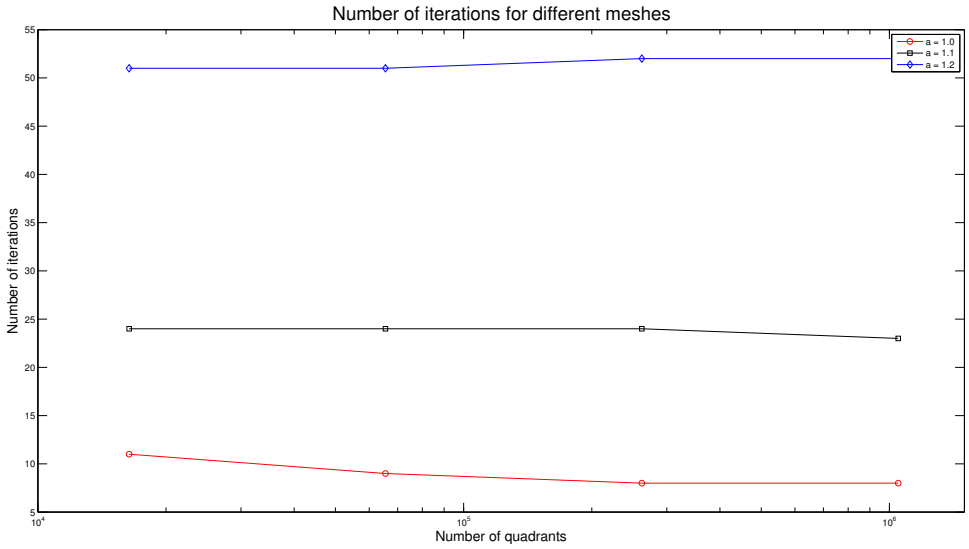


Figure 1.12: Number of iterations of PCG with the two scale preconditioner for an interpolation degree $p = 2$ needed to reach the given tolerance on the norm of the residual as a function of the number of quadrants for meshes with quadrants more and more distorted.

number of iterations slightly decreases. For example, for $p = 6$, we need 12 iterations for 128^2 quadrants but we only do 11 iterations for 1024^2. An explanation for this phenomenon might be that when we increase the number of quadrants, we actually better separate the actions of the fine and coarse preconditioners and therefore the sum of the two is a better approximation of A^{-1}. We can mention that for $p = 8$, we only need 11 iterations to solve a system that has more than 67 millions unknowns.

A second remark we can make, as already observed when we only had the fine preconditioner, is that the number of iterations grows with the degree of the interpolation. For example, with 1024^2 quadrants, we need 8 iterations when $p = 2$ but we need to do 11 iterations when $p = 8$. As before, this can be explained by the fact that as the degree of the interpolation increases, the size of the overlap decreases. A fixed sized overlap would fix this issue.

Meshes with distorted elements

We will now look at what happens when the mesh is not regular but we have quadrants that are more and more distorted. We will use the same meshes already used in the previous section and obtained with the progression tool of GMSH (see figure 1.6 for example of such meshes).

For an interpolation of degree $p = 2$, we looked at the number of iterations needed to reach the given tolerance as we increase the number of quadrants and for elements more and more distorted. Figure 1.12 shows the results. We can see that the number of iterations stays roughly the same when we increase the number of quadrants so once again the coarse part of the preconditioner does the job it is designed to do.

We can also node that when the more we distort the quadrants the more iterations we need to reach the given tolerance. This is to be expected since we developed the fine preconditioner to work optimally on quadrants aligned with the axis. This means that the more a quadrant is distorted the less accurate is the fine preconditioner part and therefore the more iterations we need. However, even with the most distorted mesh, the number of iterations stays acceptable and as explained in the previous section, the gain of not having to solve exactly on every quadrant is huge. We also have to say that the meshes presented here have an intrinsic structure since they are generated using the progression tool in GMSH. Therefore, the errors we make with our

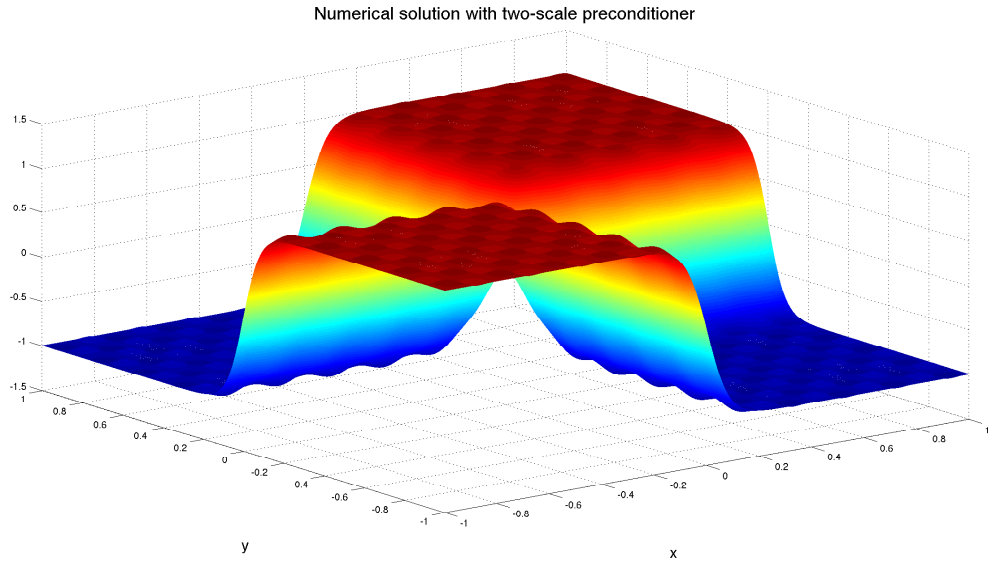


Figure 1.13: Numerical solution to problem 1.9 using an interpolation of order $p = 2$ on a non conforming mesh and obtained with PCG with the two scale preconditioner.

approximation are always in the same direction and the number of iterations needed increases. We might not have such an increase with a random mesh containing quadrants with the same quality measure.

Ajouter ici un mesh random si espace dispo

1.3.2 Influence of hanging nodes

Let us now move on to the cases when we have hanging nodes. As in the part with only the fine preconditioner, we will look at a problem where the solution has a jump so that we create hanging nodes when we use a recursive refine function in p4est. In addition to the hyperbolic tangent already presented in the previous subsection, we will include a high frequency sine wave to see how the grid adapts to capture it and how the two scale preconditioner performs in its presence.

The problem we will solve is :

$$\begin{aligned} \nabla^2 u = &-2\tanh(12x)\tanh(12y)\left[12^2(1-\tanh(12x)^2)+12^2(1-\tanh(12y)^2)\right] \\ &-10\pi^2\sin(10\pi x)\sin(10\pi y) && \text{on } \Omega \quad (1.9) \\ u = &\tanh(12x)\tanh(12y)+\frac{1}{20}\sin(10\pi x)\sin(10\pi y) && \text{on } \Gamma \quad (1.10) \end{aligned}$$

Where $\Omega = [-1;1]^2$ and Γ is the boundary. This problem has an analytic solution that is given by :

$$u(x,y) = \tanh(12x)\tanh(12y)+\frac{1}{20}\sin(10\pi x)\sin(10\pi y)$$

Figure **??** shows an example of the numerical solution computed on a non conforming mesh with PCG and the two scale preconditioner.

Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve **www.uclouvain.be/epl**