

école —
normale —
supérieure —
paris—saclay —

université
PARIS-SACLAY

Report

Master M2 in MCHPS

Project: Study and comparison of
classical DDM (Application to linear
elasticity)

Realised by: Aicha Maaoui

March 2023

ENS Paris-Saclay

Contents

0.1	Introduction	1
0.1.1	Project objectives	1
0.1.2	Problem description	1
0.2	Preliminary data setting work	1
0.2.1	Question 1.1: Analytical solution of the problem	1
0.2.2	Question 1.2: Finite element bar code (1D) with linear displacement interpolation	2
0.2.3	Question 1.3: FEA with domain decomposition methods	9
0.2.4	Notations	13
0.3	Primal approach	14
0.3.1	Question 2.1: Direct resolution of the assembled primal interface problem	14
0.3.2	Question 2.2: Sensitivity of the assembled primal Schur matrix to the parameter $\frac{h}{H}$	15
0.3.3	Question 2.3: Resolution of the assembled primal interface problem using the distributed (parallelized) conjugate gradient method	16
0.3.4	Question 2.4: Numerical scalability of distributed CG	17
0.3.5	Question 2.5: Resolution of the assembled primal interface problem by a preconditioned conjugate gradient using a Neumann preconditioner (BDD method)	18
0.3.6	Question 2.6: Conditioning of the preconditioned system	20
0.4	Dual approach	21

0.4.1	Question 2.7: Direct resolution of the assembled dual interface problem	21
0.4.2	Question 2.8: Resolution of the assembled dual interface problem using the distributed projected conjugate gradient	22
0.4.3	Question 2.9: Resolution of the assembled dual interface problem using the distributed projected conjugate gradient	23
0.5	Conclusion	23

List of Figures

1	Photo of the bar under study.	1
2	Photo of the bar under study.	3
3	Results of the bar discretization to 1 element using <i>Matlab</i>	6
4	Element stiffness matrices.	6
5	Assembled global stiffness matrix.	7
6	Results of the bar discretization to 3 element using <i>Matlab</i>	8
7	Decomposition of the bar into 3 subdomains, each having 2 finite elements.	9
8	Results of the reorganized stiffness matrix for each subdomain (using <i>Matlab</i>).	10
9	Results of the primal Schur complement for each subdomain (using <i>Matlab</i>).	11
10	Results of the right-hand side for each subdomain (using <i>Matlab</i>).	12
11	Results of the rigid body mode for each subdomain (using <i>Matlab</i>).	13
12	Results of the rigid body mode for each subdomain (using <i>Matlab</i>).	13
13	Direct resolution of the assembled primal interface problem.	14
14	Evolution of the conditioning of assembled S_p and K in function of the parameter (h/H)	15
15	Evolution of the conditioning of assembled S_p in function of the parameter (h/H)	16
16	Resolution of the assembled primal interface problem using distributed CG.	17
17	Evaluation of the time and number of iterations in function of N_s	17
18	Evolution of the number of iterations in function of the length of subdomain H . Legend: (a) $N_s = 60$, $N_s = 100$	18

19	Resolution of the assembled primal interface problem using the preconditioned CG.	19
20	Scalability results for the preconditioned CG for the primal approach. Legend: (Left) Evolution of the number of iterations in function of H , (Right) Evolution of the computational time in function of the subdomains number N_s	20
21	Resolution of the assembled dual interface problem using the direct method.	21
22	Resolution of the assembled dual interface problem using <i>FETI</i>	22
23	Scalability study of the textitFETI algorithm.	22

0.1 Introduction

0.1.1 Project objectives

The project aims at analyzing different methods of domain decomposition, i.e. the primal, dual and mixed approaches are described for a given linear elastic problem, as well as the problem resolution using an iterative method. The code scalability will be analyzed.

The commercial software *Matlab* is used to implement the various codes of the project.

0.1.2 Problem description

In this report, an academical example illustrating a bar of length L and a section S having an elastic behaviour is considered, with E the Young's modulus. It is subject to a tensile force F_d at its extremity $x = L$, whereas the displacement at its other extremity $x = 0$ is null (fixed bar). The bar is shown in figure (1).

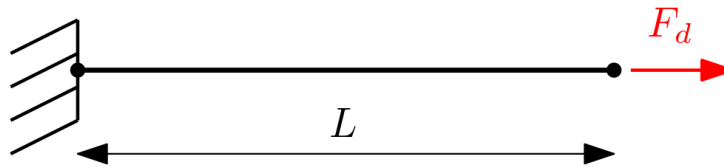


Figure 1: Photo of the bar under study.

0.2 Preliminary data setting work

0.2.1 Question 1.1: Analytical solution of the problem

Considering the case study of figure (1), the stress can be calculated using the following equation:

$$\sigma = \frac{F_d}{S} \quad (1)$$

Considering an elongation of $\Delta L = L_{new} - L$, the strain reads:

$$\epsilon = \frac{\Delta L}{L} \quad (2)$$

Taking into account the Hooke's equation described in equation (3), the strain in equation (2) can be rewritten as given in equation (4).

$$\sigma = E \epsilon \quad (3)$$

$$\boxed{\epsilon = \frac{\sigma}{E} = \frac{F_d}{E S}} \quad (4)$$

Since the displacement occurs according the x-direction only, one can only consider the deformation component ϵ_x such as:

$$\epsilon = \frac{du}{dx} \quad (5)$$

The integration of the displacement in equation (5) according to the x-direction leads to:

$$u(x) = \int_{x=0}^{x=L} \frac{F_d}{E S} dx = \frac{F_d}{E S} x + cst \quad (6)$$

where: cst is the integration constant, determined using the boundary condition at $x = 0$. Since the bar is fixed at its extremity $x = 0$, its displacement $u(x = 0)$ is null. However, from equation (6), we have $u(x = 0) = cst$, which means that: $cst = 0$. In conclusion, the displacement $u(x)$ reads:

$$\boxed{u(x) = \frac{F_d}{E S} x} \quad (7)$$

At the bar extremities $x = 0$ and $x = L$, the displacements are equal to:

$$\left\{ \begin{array}{l} u(x = 0) = 0 \\ u(x = L) = \frac{F_d L}{E S} \end{array} \right.$$

0.2.2 Question 1.2: Finite element bar code (1D) with linear displacement interpolation

In this question, the 1D bar will be divided into an input number of elements N (given by the user). The code is implemented using *Matlab*. Two case studies, where the bar

consists of only 1 element in the first one and of 3 elements in the second one, will be analyzed. For instance, figure (2) illustrates a bar of length L divided into 3 elements and having 4 nodes.

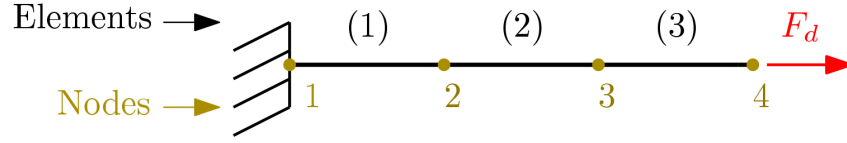


Figure 2: Photo of the bar under study.

The problem parameters considered in the following examples are presented in table (1).

Parameter	Signification	Value
S	Surface of the bar	600.10^{-6} m^2
L	Length of the bar	1 m
E	Young s modulus	210.10^9 Pa
F_d	Tensile force	1000 N

Table 1: Problem parameters.

Case 1: Bar divided into 1 element

Let us consider in the first example the bar as one element and two nodes, as shown in figure ((1)). To find the displacement, strain, stress and reaction force R_1 at node 1, the following steps are described hereunder.

Step 1: Element stiffness matrix formulation

The stiffness matrix of one element reads:

$$\underline{k}^{(e)} = \frac{S E}{L} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \quad (8)$$

Since we only have one element, the element stiffness matrix $\underline{k}^{(e)}$ also represents the global stiffness matrix \underline{K} .

Step 2: Global force vector

The global force vector reads:

$$\underline{\mathbf{F}} = \begin{bmatrix} f_1 \\ f_2 \end{bmatrix} = \begin{bmatrix} 0 \\ F_d \end{bmatrix} \quad (9)$$

Step 3: Nodal displacement

The nodal displacement vector reads:

$$\underline{\mathbf{D}} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} 0 \\ u_2 \end{bmatrix} \quad (10)$$

Step 4: Assembly of the global stiffness matrix and nodal displacement

Since the bar is regarded as one element, the global stiffness matrix and nodal displacement are equal to those of an element in equation (8) and (10). The global system to be solved is given in equation (11).

$$\boxed{\mathbf{K} \underline{\mathbf{D}} = \underline{\mathbf{F}}} \quad (11)$$

Step 5: Boundary conditions addition (Elimination approach)

The bar is fixed at its extremity $x = 0$, and thus the displacement at node 1 is null ($u_1 = 0$). Therefore, we can eliminate the first line and first row of the global stiffness matrix and the first rows of the displacement and force vectors. This is better expressed in equation (12).

$$\frac{S E}{L} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} \emptyset \\ u_2 \end{bmatrix} = \begin{bmatrix} \emptyset \\ F_d \end{bmatrix} \Rightarrow \frac{S E}{L} u_2 = F_d \quad (12)$$

The above equation gives a displacement: $u_2 = \frac{F_d L}{S E}$. We find the same results as in question (1.1).

Strain calculation

Before calculating the strain, one should express the vector of nodal shape functions $\underline{\mathbf{N}}$

(at nodes 1 and 2), as given in equation (13).

$$\underline{\mathbf{N}} = [N_1(x) \ N_2(x)] = \left[\frac{L-x}{L} \ \frac{x}{L} \right] \quad (13)$$

The strain-displacement vector $\underline{\mathbf{B}}$ is calculated as follows:

$$\underline{\mathbf{B}} = \frac{d\underline{\mathbf{N}}}{dx} = \frac{1}{L} [-1 \ 1] \quad (14)$$

The strain ϵ is then calculated using equations (10) and (14), and reads:

$$\epsilon = \underline{\mathbf{B}} \underline{\mathbf{D}} = \frac{F_d}{S E} \quad (15)$$

We obtain the same result as in equation (4) of question (1.1) found analytically.

Stress calculation

The stress is calculated using equation (16).

$$\sigma = E \epsilon = E \underline{\mathbf{B}} \underline{\mathbf{D}} = \frac{F_d}{S} \quad (16)$$

The results is equivalent to equation (1) of question (1.1).

Reaction force at node 1 calculation

The reaction force R_1 is deduced by solving the global system in equation (17).

$$\underline{\mathbf{K}} \underline{\mathbf{D}} = \underline{\mathbf{F}} + \underline{\mathbf{R}} \quad (17)$$

Where $\underline{\mathbf{R}}$ is the reaction vector equal to $\begin{bmatrix} R_1 \\ 0 \end{bmatrix}$. One can conclude that $R_1 = -\frac{SE}{L}$.

Matlab verification

The generalized code version of question (1.2) is implemented using *Matlab*. By setting $N = 1$, and using the parameters listed in table (1), the following results are obtained.

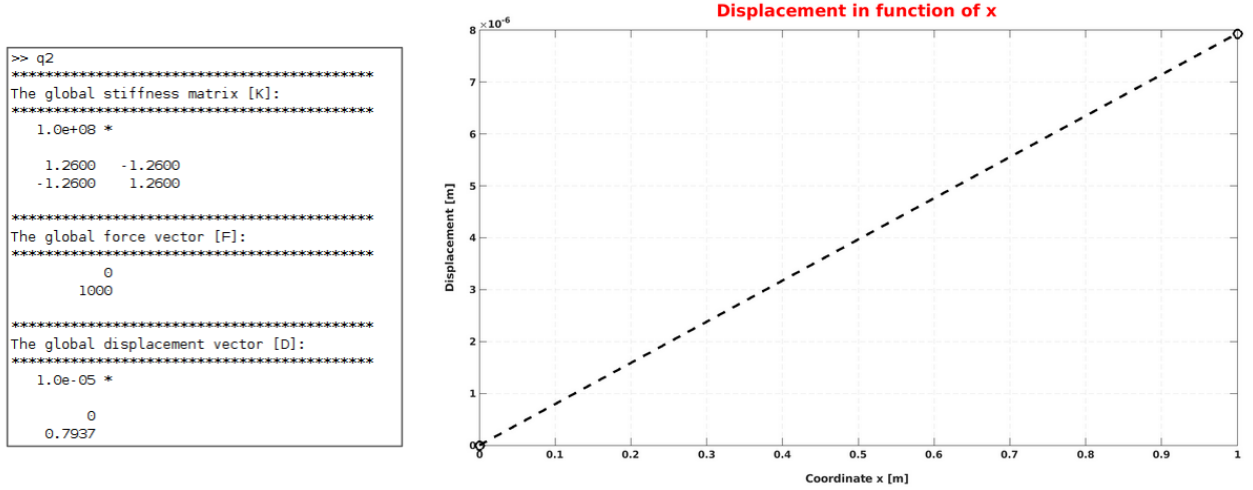


Figure 3: Results of the bar discretization to 1 element using *Matlab*.

Case 2: Bar divided into 3 element

In this section, the bar is divided into $N = 3$ elements, as illustrated in figure (2). The displacement can be calculated as presented in the steps hereunder.

Step 1: Table of connectivity

The table of connectivity (2) illustrates the elements and corresponding nodes of the structure. This helps to identify the position of the element stiffness matrix in the global one.

Element	Node i	Node j
1	1	2
2	2	3
3	3	4

Table 2: Table of connectivity.

Step 2: Element stiffness matrix

The stiffness matrices of the 3 elements are shown in figure (4), where the concerned nodes are indicated in all matrices in red.

$$k^{(e_1)} = \frac{S E}{h} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \begin{matrix} 1 \\ 2 \end{matrix} \quad k^{(e_2)} = \frac{S E}{h} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \begin{matrix} 2 \\ 3 \end{matrix} \quad k^{(e_3)} = \frac{S E}{h} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \begin{matrix} 3 \\ 4 \end{matrix}$$

Figure 4: Element stiffness matrices.

Step 3: Assembled global stiffness matrix

The assembled stiffness matrix K is deduced from figure (4) by putting each element stiffness matrix in its exact position, as illustrated in figure (5).

$$\underline{\mathbf{K}} = \frac{S E}{h} \begin{bmatrix} \overset{1}{1} & \overset{2}{-1} & 0 & 0 \\ -1 & \overset{2}{2} & -1 & 0 \\ 0 & -1 & \overset{3}{2} & -1 \\ 0 & 0 & -1 & \overset{4}{1} \end{bmatrix} \begin{matrix} \overset{1}{1} \\ \overset{2}{2} \\ \overset{3}{3} \\ \overset{4}{4} \end{matrix}$$

Figure 5: Assembled global stiffness matrix.

Step 4: Global force vector

The tensile force is only applied at node 4, and thus the global force vector $\underline{\mathbf{F}} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ F_d \end{bmatrix}$

Step 5: Global displacement vector

The global nodal displacement vector $\underline{\mathbf{D}} = \begin{bmatrix} 0 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}$, since the bar is fixed at its node 1 ($u_1 = 0$). It is solved using equation (11).

Step 6: Strain calculation

The nodal shape functions vector $\underline{\mathbf{N}}$ for each element is given hereunder.

$$\begin{cases} N^{(e_1)} &= [(1 - \frac{x}{h}) \ (\frac{x}{h})] \\ N^{(e_2)} &= [(2 - \frac{x}{h}) \ (\frac{x}{h} - 1)] \\ N^{(e_3)} &= [(3 - \frac{x}{h}) \ (\frac{x}{h} - 2)] \\ N^{(e_4)} &= [(4 - \frac{x}{h}) \ (\frac{x}{h} - 3)] \end{cases}$$

The element strain vector is given using the following equation:

$$\underline{\epsilon}_i = \frac{1}{h} \begin{bmatrix} -1 & 1 \end{bmatrix} \begin{bmatrix} u_i \\ u_{i+1} \end{bmatrix} \quad (18)$$

Where: $\underline{\epsilon}$ is the vector of size $(N \times 1)$ containing the strain of all elements of the bar, and i is an index varying from 1 to N .

Step 7: Stress calculation

The stress vector of size $(N \times 1)$ is deduced from the strain in equation (18), given that:

$$\underline{\sigma}_i = E \underline{\epsilon}_i \quad (19)$$

Where i is an index varying from 1 to N .

Matlab verification

The generalized code version of question (1.2) is implemented using *Matlab*. By setting $N = 3$, and using the parameters listed in table (1), the following results are obtained (as given in figure (6)).

```
>> q2
*****
The global stiffness matrix [K]:
*****
    378000000    -378000000         0         0
    -378000000     756000000    -378000000         0
         0    -378000000     756000000    -378000000
         0         0    -378000000     378000000
*****
The global force vector [F]:
*****
         0
         0
         0
        1000
*****
The global displacement vector [D]:
*****
    1.0e-05 *
         0
        0.2646
        0.5291
        0.7937
```

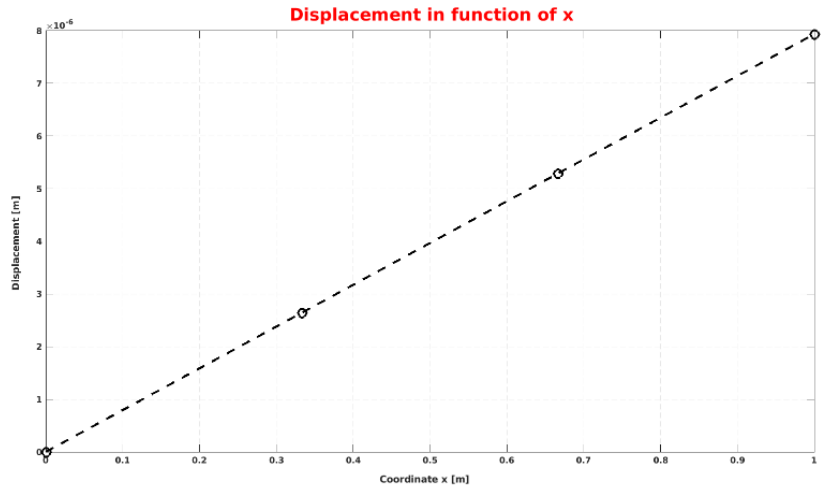


Figure 6: Results of the bar discretization to 3 element using *Matlab*.

0.2.3 Question 1.3: FEA with domain decomposition methods

In this question, the problem of the 1D bar (presented in figure (1)) is solved using the domain decomposition method. To do so, the bar is divided into a number of subdomains of length H each. Every subdomain is composed of a set of finite element, each having a length equal to h . For the sake of comparison between analytical and numerical resolution, we define the following:

- The number of subdomains is equal to 3,
- The number of finite elements is equal to 2,
- The number of nodes in each subdomain is equal to 3.

The nodes are divided to internal and boundary nodes, as illustrated in figure (7). One should underline that both the first node ($x = 0$) and last node ($x = L$) are considered internal since the boundary conditions are taken into account locally at the subdomain level. Moreover, we have two interfaces between three subdomains.

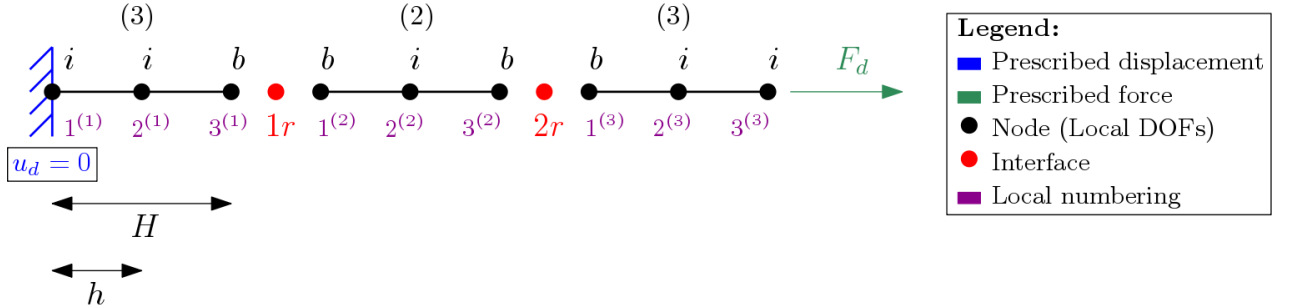


Figure 7: Decomposition of the bar into 3 subdomains, each having 2 finite elements.

Description of the corresponding implementation

The domain decomposition as well as the definition of the operators for each subdomain is implemented in the function ***decompose_primal_dual***. This function takes in argument the desired number of subdomains N_s and the number of elements per subdomain. It takes also the total length and section surface of the bar, the Young's modulus and the applied force. A map was defined to relate the local numbering of the nodes with the global numbering. The interface nodes were also defined. The primal and

dual assembly matrices were defined for each subdomain. Next, the stiffness matrix and force vector of each subdomain were computed by assembling the elementary matrices of the elements. For the first subdomain, the imposed displacement at the first node was taken into account using the elimination approach (an other approximation method could be used instead of elimination which is the penalty method, but will not be taken into account in this work). The stiffness matrices and force vectors of all the substructures were reordered in order to have the internal nodes before the boundary nodes. Finally, the primal Schur complement, the dual Schur complement and the rigid body modes are computed. These outputs will be used as inputs for the different solving approaches implemented in the following.

Results discussion

Stiffness Matrix for each subdomain S_p

Since we have 2 elements in every subdomain, the stiffness matrix for every subdomain is the same as in figure (5). Let's denote $k_0 = ES/h$, the analytical resolution of the reordered stiffness matrix for each subdomain, taking into account the Dirichlet boundary condition of the first node by the elimination approach is given in equation (20).

$$K^{(1)} = k_0 \begin{pmatrix} 2 & -1 \\ -1 & 1 \end{pmatrix}; \quad K^{(2)} = k_0 \begin{pmatrix} 2 & -1 & -1 \\ -1 & 1 & 0 \\ -1 & 0 & 1 \end{pmatrix}; \quad K^{(3)} = k_0 \begin{pmatrix} 2 & -1 & -1 \\ -1 & 1 & 0 \\ -1 & 0 & 1 \end{pmatrix} \quad (20)$$

The reordered stiffness matrices have been computed for every subdomain using *Matlab*, taking into account the parameters of table (1), as illustrated in figure (8).

Ksub {1, 1}		
	1	2
1	1.5120e+09	-756000000
2	-756000000	756000000

Ksub {1, 2}			
	1	2	3
1	1.5120e+09	-756000000	-756000000
2	-756000000	756000000	0
3	-756000000	0	756000000

Ksub {1, 3}			
	1	2	3
1	1.5120e+09	-756000000	-756000000
2	-756000000	756000000	0
3	-756000000	0	756000000

Figure 8: Results of the reorganized stiffness matrix for each subdomain (using *Matlab*).

Primal Schur Complement for each subdomain S_p

If we consider a subdomain (s) loaded on internal degree of freedom, the local equilibrium reads:

$$\begin{pmatrix} K_{ii}^{(s)} & K_{ib}^{(s)} \\ K_{bi}^{(s)} & K_{bb}^{(s)} \end{pmatrix} \begin{pmatrix} u_i^{(s)} \\ u_b^{(s)} \end{pmatrix} = \begin{pmatrix} f_i^{(s)} \\ f_b^{(s)} \end{pmatrix} \quad (21)$$

Then, the condensation of the equilibrium on the interface reads:

$$S_p^{(s)} u_b^{(s)} = b_p^{(s)} \quad (22)$$

Where S_p is the primal Schur complement equal to :

$$S_p^{(s)} = K_{bb}^{(s)} - K_{bi}^{(s)} K_{ii}^{(s)-1} K_{ib}^{(s)} \quad (23)$$

The analytical solution of the primal Schur complement for each subdomain are given in equation (24) and considering equations (20) , (21) and (23).

$$S_p^{(1)} = \frac{k_0}{2}; \quad S_p^{(2)} = \frac{k_0}{2} \begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}; \quad S_p^{(3)} = 0 \quad (24)$$

The implementation results of the primal Schur complement for each subdomain in *Matlab* is shown in figure (9).

SPsub		1x3 cell				
SPsub{1, 1}		SPsub{1, 2}		SPsub{1, 3}		
	1		1	2		1
1	378000000	1	378000000	-378000000	1	0
		2	-378000000	378000000		

Figure 9: Results of the primal Schur complement for each subdomain (using *Matlab*).

Right-hand side for each subdomain b_p

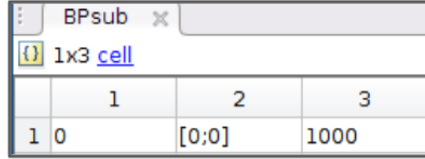
The right-hand side or the effort imposed on the substructure $b_p^{(s)}$ is given in equation (25) as follows.

$$b_p^{(s)} = f_b^{(s)} - K_{bi}^{(s)} K_{ii}^{(s)-1} f_i^{(s)} \quad (25)$$

The analytical solution of the right-hand side for each subdomain are given in equation (26) and considering equations (20) , (21) and (25).

$$b_p^{(1)} = 0; \quad b_p^{(2)} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}; \quad b_p^{(3)} = F_d \quad (26)$$

The implementation results of the right-hand side for each subdomain in *Matlab* is shown in figure (10).



	1	2	3
1	0	[0;0]	1000

Figure 10: Results of the right-hand side for each subdomain (using *Matlab*).

Rigid body modes each subdomain bp

The rigid body modes are calculated as follows:

$$R_b = \ker(S_p) \quad (27)$$

Since there is no rigid body mode for the first node, the rigid body modes for subdomains (2) and (3) are given in equation (28).

$$R_b^{(1)} = 0; \quad R_b^{(2)} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}; \quad R_b^{(3)} = 1 \quad (28)$$

The implementation results of the rigid body mode for each subdomain in *Matlab* is shown in figure (11).

	1	2	3
1	[1]	[1;1]	1

Figure 11: Results of the rigid body mode for each subdomain (using *Matlab*).

Assembly operators A and \underline{A}

The assembly operators A (primal) and \underline{A} (dual) are given hereunder.

Primal assembly operator:

The primal assembly operator of the three subdomains are given in equation (29).

$$A^{(1)} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad A^{(2)} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad A^{(3)} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \quad (29)$$

Dual assembly operator:

The dual assembly operator of the three subdomains are given in equation (30).

$$\underline{A}^{(1)} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \quad \underline{A}^{(2)} = \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix} \quad \underline{A}^{(3)} = \begin{pmatrix} 0 \\ -1 \end{pmatrix} \quad (30)$$

The implementation results of the assembly operators A and \underline{A} using *Matlab* are shown in figure (12).

	1	2	3
1	[1;0]	[1,0;0,1]	[0;1]

	1	2	3
1	[1;0]	[-1,0;0,1]	[0;-1]

Figure 12: Results of the rigid body mode for each subdomain (using *Matlab*).

0.2.4 Notations

In the following sections, the following notations are used :

- Bold notations for assembled operators,
- superscript \diamond for block notations.

0.3 Primal approach

0.3.1 Question 2.1: Direct resolution of the assembled primal interface problem

The assembled primal interface problem is given as follows:

$$\mathbf{S}_p \mathbf{u}_b = \mathbf{b}_p \quad (31)$$

where :

$$\mathbf{S}_p = A^\diamond S_p^\diamond A^T : \text{the assembled primal Schur complement} \quad (32)$$

$$\mathbf{b}_p = A^\diamond b_p^\diamond : \text{the assembled right-hand side vector} \quad (33)$$

Using the direct method in *Matlab*, such as $\mathbf{u}_b = \mathbf{S}_p \backslash \mathbf{b}_p$ gives the solution illustrated in figure (13) (implementation in the script "*primalSchur*" in *Matlab*).

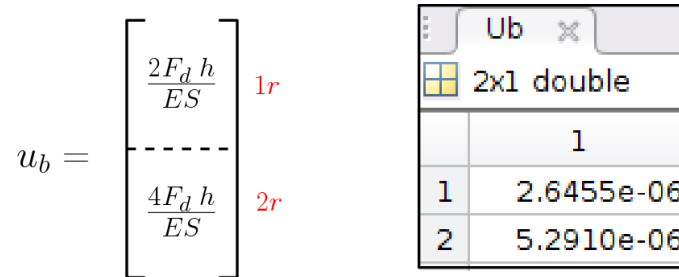


Figure 13: Direct resolution of the assembled primal interface problem.

This result is in agreement with the analytical solution of equation (7), by considering $x = 2 \times h$ and $x = 4 \times h$.

It should also be noted that scripts "*decompose_primal_dual*", "*rigid_body_mode*" and "*assemble*" give the primal and dual Schur complements, rigid body modes, assembly operators and needed parameters for primal and dual approaches. They are called in the *main* script under the name "(Q1.3) : Primal and Dual subdomain decomposition".

0.3.2 Question 2.2: Sensitivity of the assembled primal Schur matrix to the parameter $\frac{h}{H}$

This question is implemented in the section *Evaluation of the conditioning of the main script*. We have $h = H/N$ with N the number of elements in each subdomain. Thus, for a fixed number of subdomains, studying the sensitivity to h/H is equivalent to studying the sensitivity to $1/N$. The conditioning of the stiffness matrix of the unstructured problem was also computed with total number of elements equal to the elements. The *Matlab* function *eig* was used to compute the eigenvalues of the matrices.

We fixed $N_s = 10$ and N was varied from 2 to 50. Figures (14) and (15) show the obtained results. We can notice that when h becomes small, the conditioning of K rises exponentially. In the other hand, for a domain with the same total number of elements, the conditioning of assembled primal Schur matrix remain much lower, up to 3 order of magnitude lower. Figure (15) shows when h/H becomes small, the conditioning of S_p becomes instable and starts to oscillate.

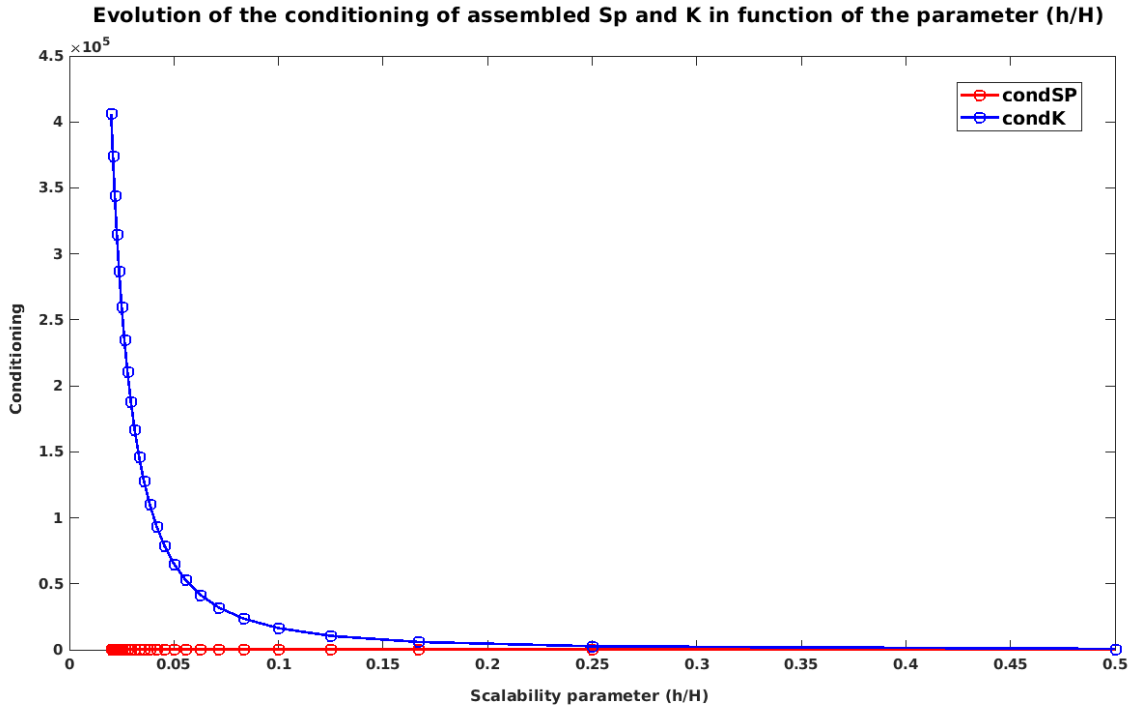


Figure 14: Evolution of the conditioning of assembled S_p and K in function of the parameter (h/H) .

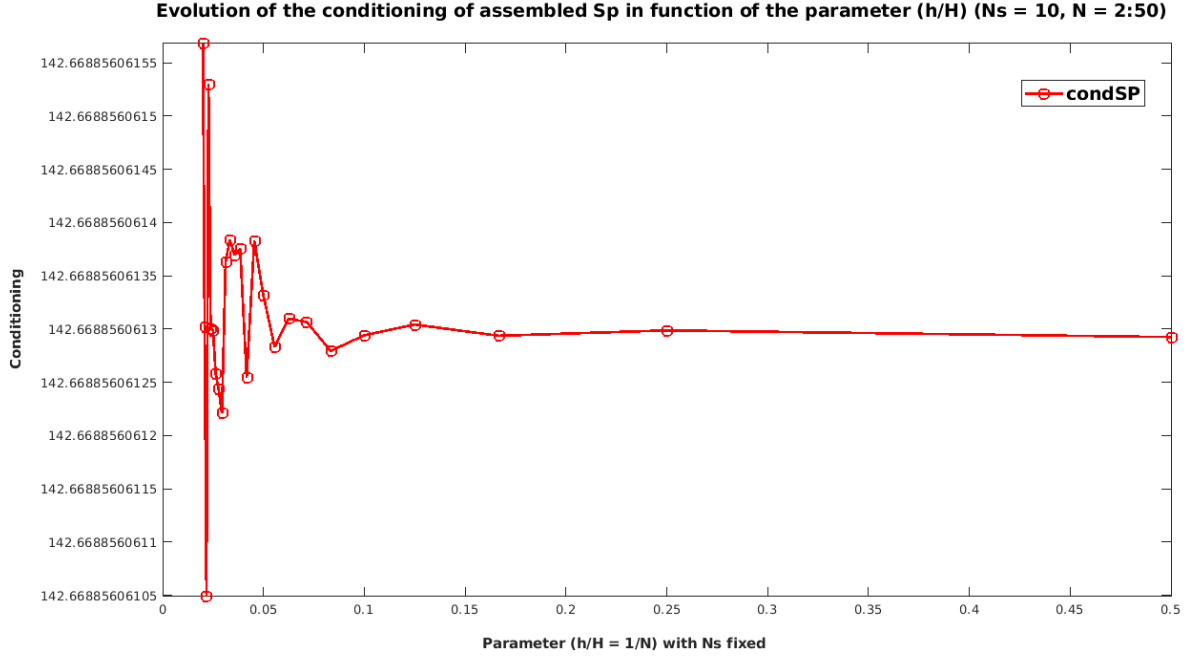


Figure 15: Evolution of the conditioning of assembled S_p in function of the parameter (h/H) .

0.3.3 Question 2.3: Resolution of the assembled primal interface problem using the distributed (parallelized) conjugate gradient method

In this question, the algorithm of the distributed conjugate gradient in document **Conjugate gradient algorithm to solve the primal interface problem** available in the **References** section is implemented using *Matlab*. The algorithm is adapted (step of reorthogonalization and updating search direction) in a way to consider the re-orthogonalized conjugate gradient version (algorithm 3 in document **Krylov iterative solvers** in the **References** section). The reorthogonalization is implemented (over the orthogonalization) because it can be better by helping to maintain the orthogonality of the search directions, which is important for the algorithm convergence. Moreover, the convergence criteria is added in order to assure the convergence of the iterative solver :

$$\frac{\|rb\|}{\|rb_0\|} < \epsilon$$

The result of the displacement u_b is compared to the direct solver in figure (13) and analytical resolution, which leads to a great agreement as illustrated in figure (16).

This question is implemented in the "*CG_primal_interface*" script in *Matlab*.

Ub_iter	
2x1 double	
	1
1	2.6455e-06
2	5.2910e-06

Figure 16: Resolution of the assembled primal interface problem using distributed CG.

0.3.4 Question 2.4: Numerical scalability of distributed CG

To evaluate the numerical scalability of the distributed gradient solution, one has to consider a constant parameter $\frac{h}{H} = \frac{1}{N}$. We fix a number of elements $N = 3$ per subdomain and we increase the number of subdomains. Figure (17) illustrates the evolution of both the number of iterations and computational time with respect to the number of subdomains, which is increasing as we increased the size of the computational domain.

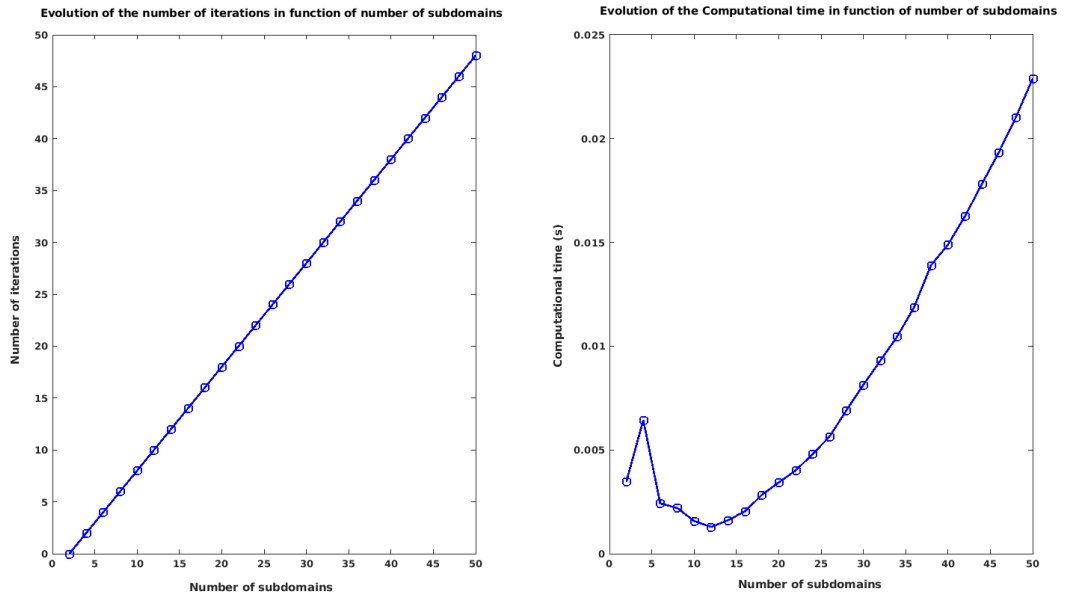


Figure 17: Evaluation of the time and number of iterations in function of N_s .

Figure (18) illustrates the evolution of the number of iterations in function of the length of subdomain H for $N_s = 60$ and $N_s = 100$, respectively. One can conclude the following:

- The primal Schur algorithm is numerically extensible, i.e.; it can handle a large number of subdomains N_s ,
- The primal Schur algorithm is numerically scalable if $H > 0.25m$, i.e.; the number of iterations to converge does not depend very much on H if superior to $0.25m$

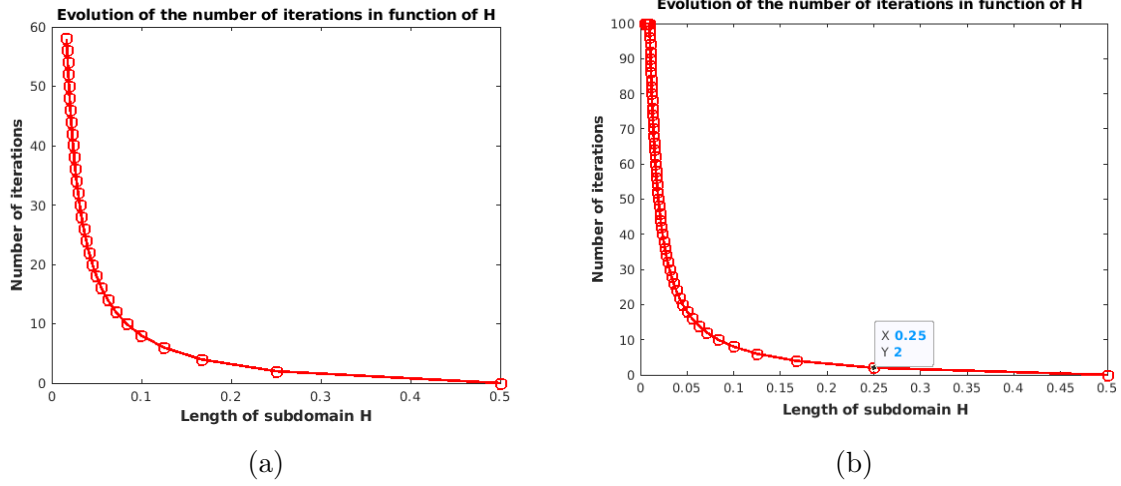


Figure 18: Evolution of the number of iterations in function of the length of subdomain H . **Legend:** (a) $N_s = 60$, $N_s = 100$.

This question is implemented in the *main* script in *Matlab*, under the name "(Q2.4) : Numerical Scalability of distributed CG".

0.3.5 Question 2.5: Resolution of the assembled primal interface problem by a preconditioned conjugate gradient using a Neumann preconditioner (BDD method)

The preconditioner for the primal Schur interface (also called *BDD*) is given in equation (34) as follows:

$$\tilde{\mathbf{S}}_p^{-1} = \sum_s \mathbf{M}^{(s)} \mathbf{A}^{(s)} \mathbf{S}_d^{(s)} \mathbf{A}^{(s)T} \mathbf{M}^{(s)} \quad (34)$$

For homogeneous structures, $\mathbf{M}^{(s)} = \text{diag}\left(\frac{1}{\text{multiplicity}}\right)$. In our case, the multiplicity in our case is equal to 2.

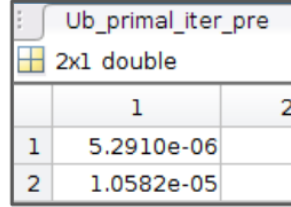
The implemented assembled preconditioner of the primal interface problem in *Matlab* (function in the script: "*preCG_BDD.m*") reads:

$$\tilde{\mathbf{S}}_p^{-1} = \tilde{\mathbf{A}} \tilde{\mathbf{S}}_p^{+\diamond} \tilde{\mathbf{A}}^T \quad (35)$$

where:

- $\tilde{S}_p^{s+} = S_d^s$: The dual Schur complement
- $\tilde{A} = (A^\diamond M^\diamond A^{\diamond T})^{-1} A^\diamond M^\diamond$: The scale assembly operator
- $M^\diamond = I_d$: For homogeneous structures (our case).

The result of the displacement u_b using the preconditioned CG is illustrated in figure (19). One can see that comparing to the previous results displacement, there the displacement obtained hereunder is multiplied by a factor of 2 (investigated, but no explanation found).



	1	2
1	5.2910e-06	
2	1.0582e-05	

Figure 19: Resolution of the assembled primal interface problem using the preconditioned CG.

Scalability study of the algorithm

Both the evolution of the number of iterations as a function of H and the computational time as a function of the number of subdomains N_s have been plotted using *Matlab* for maximum number of $N_s = 200$, as shown in figure (20).

The left figure in (20) shows that the number of iterations is constant in function of H . Thus, the preconditioned conjugate gradient in the primal approach is numerically scalable. Moreover, the computational time in the case of the distributed CG in figure (17) reaches a maximum of almost $0.024s$ for $N_s = 50$, however the preconditioned CG has a computational time inferior to $0.005s$, making this latter faster.

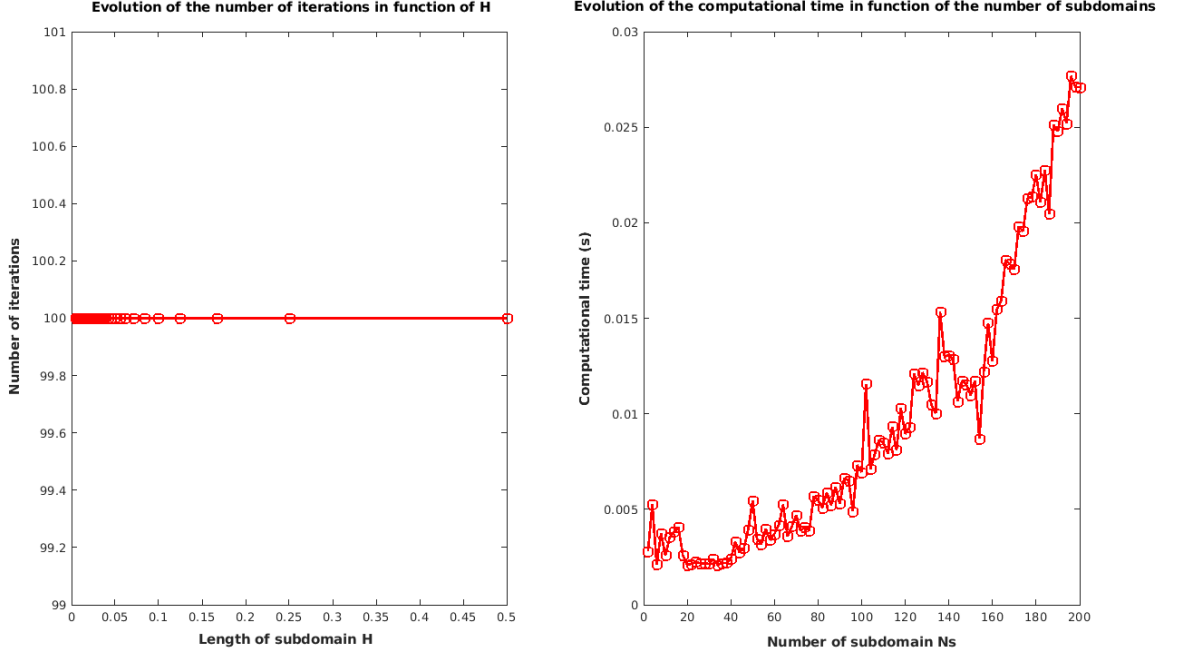


Figure 20: Scalability results for the preconditioned CG for the primal approach. **Legend:** (Left) Evolution of the number of iterations in function of H , (Right) Evolution of the computational time in function of the subdomains number N_s .

0.3.6 Question 2.6: Conditioning of the preconditioned system

The conditioning of $\tilde{\mathbf{S}}_p^{-1} \mathbf{S}_p$ was computed. Contrary to what was expected, the computed conditioning was equal to about 40. This unexpected high value is probably related to a wrong implementation of the preconditioner, which might also explain the factor of 2 found on the displacement of the interface nodes.

I am totally aware that there is a problem in the implementation of the *Matlab* function "*preCG_BDD.m*". I still need to better handle the preconditioner implementation, in both assembled and distributed formulations.

0.4 Dual approach

0.4.1 Question 2.7: Direct resolution of the assembled dual interface problem

The solution of the dual interface problem reads:

$$\begin{pmatrix} \mathbf{S}_d & \mathbf{G} \\ \mathbf{G}^T & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{\lambda}_b \\ \alpha^\diamond \end{pmatrix} = \begin{pmatrix} -\mathbf{b}_d \\ -e^\diamond \end{pmatrix} \quad (36)$$

where:

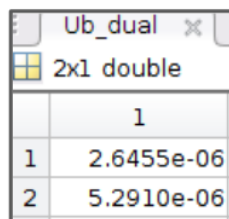
$$\mathbf{S}_d = \underline{\mathbf{A}}^\diamond \mathbf{S}_d^\diamond \underline{\mathbf{A}}^{\diamond T} \quad (37)$$

$$\mathbf{b}_d = \underline{\mathbf{A}}^\diamond \mathbf{b}_d^\diamond \quad (38)$$

$$\mathbf{G} = \underline{\mathbf{A}}^\diamond \mathbf{R}_b^\diamond \quad (39)$$

$$e^\diamond = \mathbf{R}_b^{\diamond T} \mathbf{b}_p^\diamond \quad (40)$$

The direct giving both \mathbf{b}_d and α^\diamond has been implemented in *dualSchur.m* in *Matlab*. Its execution leads to the interface displacement u_b shown in figure



Ub_dual	
2x1 double	
	1
1	2.6455e-06
2	5.2910e-06

Figure 21: Resolution of the assembled dual interface problem using the direct method.

Note : The plot of the full displacement solution (U_i and U_b) for both primal and dual approaches using algorithms are possible and implemented inside the *plot_U* script of *Matlab*.

0.4.2 Question 2.8: Resolution of the assembled dual interface problem using the distributed projected conjugate gradient

This method is also called Finite Element Tearing and Interconnecting (FETI) method in the literature. The constrained Kyrlov method with projector implementation is used to solve the above system (equation (36)). Both the distributed version of the projected CG (script "*FETI_PCPG.m*") and the assembled version (script "*FETI_PCPG_bis.m*") have been implemented in *Matlab*. Both leading to the same result of the displacement at the interface u_b , as illustrated in figure (22).

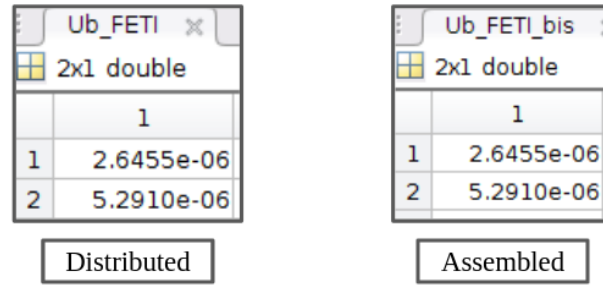


Figure 22: Resolution of the assembled dual interface problem using *FETI*.

Evaluation of scalability and computational time

The algorithm scalability and computational time of both the distributed and assembled dual algorithms are plotted in figure (23).

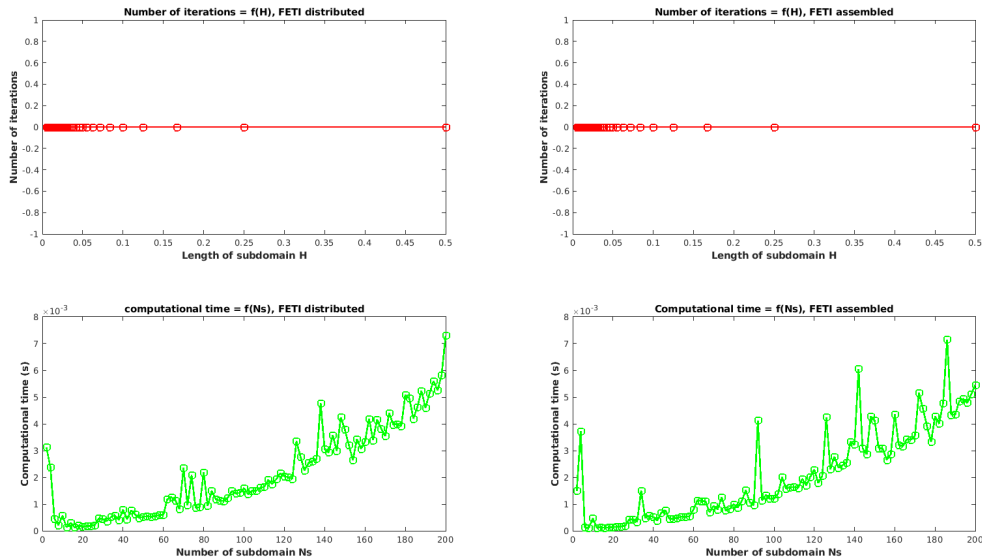


Figure 23: Scalability study of the textitFETI algorithm.

One can conclude that the FETI algorithm to solve the dual approach is numerically scalable. Moreover, the comparison of the computational results of figures (23) and figure (20) shows that the FETI algorithm is faster than the preconditioned CG to solve the primal problem since for $N_s = 200$ subdomains, the elapsed time for the dual is less than $8e^{-03}s$ lower than $0.03s$.

0.4.3 Question 2.9: Resolution of the assembled dual interface problem using the distributed projected conjugate gradient

At convergence, we have:

$$R_b^{(s)} \lambda_b^{(s)} = 0 \quad (41)$$

Using the previous equation one can compute the rigid body modes at convergence.

As in the primal interface problem, preconditioner can be used here to improve the algorithm. It should be noted that only the dual Schur matrix need to preconditioned not all the Lagrangian matrix. We used here the Dirichlet preconditioner given by:

$$\tilde{S}_d^{-1} = \sum_s M^{(s)} \underline{A}^{(s)} S_p^{(s)} \underline{A}^{(s)T} M^{(s)} \quad (42)$$

where $M^{(s)}$ is the same as the one used in the preconditioner to the primal interface problem. The preconditioned dual algorithm is known in the literature as the FETI Preconditioned Conjugate Projected Gradient (FETI PCPG).

0.5 Conclusion

In this report, the primal and dual approaches have been investigated and implemented using *Matlab*. Then, the resolution of the assembled primal and dual interface problems have been performed, as well as their numerical scalability. Moreover, it has been concluded that the implementation FETI method can be further improved [4].

References

- [1] Farhat, C. (1991). A Lagrange multiplier based divide and conquer finite element algorithm. *Computing Systems in Engineering*, 2(2-3), 149-156.
- [2] Rixen, D. J., & Farhat, C. (1999). A simple and efficient extension of a class of substructure based preconditioners to heterogeneous structural mechanics problems. *International Journal for Numerical Methods in Engineering*, 44(4), 489-516.
- [3] Farhat, C., Pierson, K., & Lesoinne, M. (2000). The second generation FETI methods and their application to the parallel solution of large-scale linear and geometrically non-linear structural analysis problems. *Computer methods in applied mechanics and engineering*, 184(2-4), 333-374.
- [4] Kozubek, T., Vondr1k, V., Men81k, M., Hor1k, D., Dost1l, Z., Hapla, V., ... & Čerm1k, M. (2013). Total FETI domain decomposition method and its massively parallel implementation. *Advances in Engineering Software*, 60, 14-22.