

École Normale Supérieure Paris-Saclay

M2 Modélisation et Simulation avec HPC

Projet : Étude et comparaison de DDM

– Application en élasticité linéaire –

****

Projet encadré par :

Pierre-Alain GUIDAULT

**Table des matières**

1. [Présentation du problème et résolution analytique 2](#_TOC_250013)
   1. [Définition du problème 2](#_TOC_250012)
   2. [Résolution analytique 2](#_TOC_250011)
2. [Résolution par la méthode des éléments finis 3](#_TOC_250010)
3. [Résolution par décomposition de domaine – approche primale 4](#_TOC_250009)
   1. [Principe 4](#_TOC_250008)
   2. [Code Matlab 5](#_TOC_250007)
      1. [Écriture des paramètres du problème 5](#_TOC_250006)
      2. [Conditionnement de la matrice de Schur primal et de K 7](#_TOC_250005)
      3. [Résolution du problème par la méthode du gradient conjugué parallélisé 10](#_TOC_250004)
      4. [Résolution du problème par la méthode du gradient conjugué pré-conditionné. 12](#_TOC_250003)
4. [Résolution par décomposition de domaine – approche duale 14](#_TOC_250002)
   1. [l’approche duale 14](#_TOC_250001)
   2. [résolution du système dual assemblé 14](#_TOC_250000)

**Introduction**

Dans ce projet, on s’intéresse à la résolution d’un problème mécanique simple, à savoir une barre en traction 1D, en utilisant différentes méthodes de décomposition de domaine.

Dans un premier temps, une solution analytique de référence est calculé et une résolution du problème par la méthode des éléments finis est aussi implémenter afin de pouvoir les comparer aux outils de résolution par décomposition de domaine.

Dans un second temps, deux approches – primale et duale – sont implémentées pour résoudre le problème poutre. Des méthodes directes puis parallélisées ont été testées avec succès et l’extensibilité des méthodes a également été étudiée.

# Présentation du problème et résolution analytique

## Définition du problème

Le problème mécanique à résoudre est schématisé figure 1. Il s’agit d’une barre de section constante S et de longueur L, soumise à son extrémité à un effort ponctuel *Fd* dans sa direction principale, *ex*. Ainsi, le comportement ne dépend que de la variable d’espace x et pas de y. Le comportement est supposé homogène et linéaire élastique, de module d’Young E. On cherche alors à déterminer la composante de déplacement *ux* de la poutre en tout point.

**y**

**Fd**

**x**

FIGURE 1 – Problème de barre en traction.

## Résolution analytique

On se place ici sous l’hypothèse d’Euler-Bernoulli, ce qui conduit aux équations du modèle poutre suivantes pour la résultante et le moment





*∂R* = 0*,*

*∂x*

*∂M* + *e*

*∂x*

*x*

∧ *R* = 0*.*

(1)

La première ligne conduit à une résultante d’effort constante, dont la norme est obtenue via les conditions aux limites

*R* = *Fd* = *N.* (2)

On obtient alors pour la dérivée du moment

*∂M* = 0*.* (3)

*∂x*

Dans la théorie d’Euler-Bernoulli, on a la relation suivante entre déplacement axial et résultante axiale

*N* = *ESux.* (4)

On en déduit la valeur du déplacement axial en tout point

*ux* =

1

*ES Fd.* (5)

Le moment quant à lui est obtenue par intégration et prise en compte des conditions aux limites (moment nul en *x* = *L*)

*M* = 0*.* (6)

# Résolution par la méthode des éléments finis

On décompose la poutre en éléments de longueur *h*, comme on peut le voir figure 2. On approche alors le champ de déplacement continu par un champ discrétisé, dont les valeurs s’expriment aux noeuds du maillage. Des fonctions de forme sont évaluées en un point donné, afin de donner une interpolation du déplacement à partir des valeurs nodales.

*N* +1

Σ

*u*(*x*) ≈ *uh*(*x*) = *uiφi*(*x*)*.* (7)

*i*=1

Ici, N correspond au nombre d’éléments, il y a donc N+1 noeuds. On se donne pour chaque élément des fonctions de

**y**



**x**

**Fd**

**h**

FIGURE 2 – Discrétisation EF du problème.

forme linéaires. Pour chaque élément i, on a deux fonctions de forme, *φi* et *φi*+1, dont les valeurs varient entre 0 et 1, comme schématisé figure 3. Dans le cas de tel fonctions de forme, la matrice de raideur élémentaire pour un élément i,

p**(x) 1**

p**i** p**i+1**

**0 ** **x**



**élément i**

FIGURE 3 – Fonctions de forme linéaires sur un élément. obtenue par intégration entre *xi* et *xi*+1 vaut

***K****i* = *ES* A 1 −1 B *.* (8)

*e*

*h*

−1 1

Le problème assemblé s’écrit alors

***K****u* = *F,* (9)

avec ***K*** la matrice de rigidité assemblée, *u* le vecteur des déplacements nodaux (selon x) et *F* le vecteur des efforts extérieurs. Grâce aux conditions limites, on connaît la valeur de la composante *u*1 qui est nulle, la poutre étant encastrée en *x* = 0. De même, le vecteur *F* est nul sauf sa première composante, l’effort de réaction inconnu, et sa dernière composante qui est de norme |*Fd*|. Les inconnues que l’on cherche sont donc les composantes nodales *u*2 à *uN* +1, que l’on obtient en inversant le système, après élimination de la première ligne correspondant au déplacement du noeud 1 qui est connu.

Les résultats sont tracés figure 4 pour un module d’Young de 20 *GPa*, une section de 10 *mm*2 et une longueur de 100 *mm*. On observe comme on s’y attendait une évolution linéaire du déplacement, et un champ de déplacement EF qui se superpose parfaitement à la solution analytique.

10-4 **problem FE and analytical solutions**

6

Analytical solution FE solver solution

5

4

3

2

1

0

0 10 20 30 40 50 60 70 80 90 100

FIGURE 4 – Déplacement selon x, solution analytique et solution EF.

# Résolution par décomposition de domaine – approche primale

## Principe

La méthode de Schur 1 primale porte son nom du fait que les inconnues nodales de l’interface séparant les domaines sont des variables de déplacements, c’est à dire les mêmes que celles des sous-domaines. Le problème mécanique associé est représenté sur la figure 5.

Après discrétisation du problème on trouve :

( ) ( ) = A *K*(*s*) *K*(*s*) B A *u*(*s*) B = A *f* (*s*) B

(10)



*K s*

*K s*

 ∀*s* ∈ ***S****, K*

*s*

*u*

*s*

*i*(*i* )

*bi*

*i*(*b* )

*bb*

*.*

*i*(*s*)

*ub*

*i*(*s*)

*fb*

*,*



 ∀Γ(*s,s'*)*,* (*s, s'*) ∈ ***S****, u*(*s*) − *u*(*s'*) = 0 *.*

L’ indice *i* représente les degrés de liberté (ddl) internes au sous-domaine (*s*) et l’indice *b* représente les ddl au bord par rapport à l’interface de (*s*). Pour une sous-domaine (*s*) :

*b*

*b*

A *K*(*s*) *K*(*s*) B A *u*(*s*) B = A *f* (*s*) B

*i*(*i*

*K s*)

*K s*

*bi*

*i*(*b* )

*bb*

*. i*(*s*)

*ub*

*i*(*s*) *. fb*

1. Mathématicien russe, 1875-1941

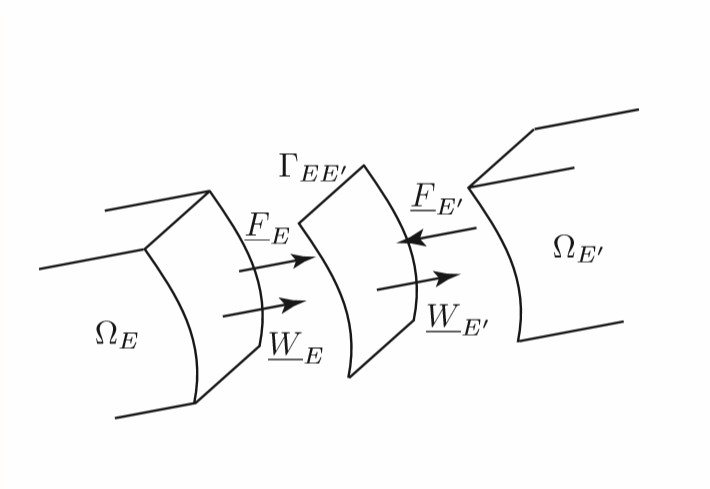


FIGURE 5 – Décomposition en deux sous-domaines.

les degrés de liberté internes des sous-domaine sont données par la première ligne comme suit :

*u*(*s*) = *K*(*s*)*−*1!*f* (*s*) − *K*(*s*)*u*(*s*)" (11)

*i*

*ii*

*i*

*ib*

*b*

En injectant cette équation dans la deuxième ligne et par élimination de Gauss de *ui* , on obtient :

!*K*(*s*) − *K*(*s*)*K*( *−*1*K*(*s*)"*u*(*s*)

= *f* (*s*) − *K*(*s*)*K*(*s*)*−*1*f* (*s*)

(12)

En notant

*bb bi ii*

*ib b*

*b bi ii* *i*

*S*(*s*) = *K*(*s*) − *K*(*s*)*K*(*s*)*−*1*K*(*s*)

*p bb bi ii ib*

*b*(*s*) = *f* (*s*) − *K*(*s*)*K*(*s*)*−*1*f* (*s*)

*p b*

On a alors le système suivant à résoudre :

*S*(*s*)*u*(*s*)

*bi ii* *i*

= *b*(*s*)

(13)

*p b p*

La matrice *S*(*s*) est appelée le complément de Schur primal. Une fois que la solution *u*(*s*) est calculée, il suft de la

*p b*

remplacer dans l’expression (11) pour avoir toutes les inconnues.

## Code Matlab

### Écriture des paramètres du problème

Les données du problème sont comme suit :

% d a t a

E=200 e9 ; % Young Modulules Pa

S=10 e −6; % s u r f a c e a r e a 10 e−6 m =10 mm ; ES=E\*S ;

L = 0 . 1 0 0 ; % 100 mm l e n g t h b a r

% Domain Decomposition

h = 0 . 0 0 5 ; % e l e m e n t s i z e 5mm

ne = 5 ; % e l e m e n t number p e r s u b s t r u c t u r e

H=h\* ne ; % subdolmain s i z e

Nss=L / H; % Number of s u b s t r u c t u r e

nd=ne + 1 ; % Nods number p e r s u b s t r u c t u r e V e c t o r \_ n o d e s = ( 0 : h : L ) ;% t o t a l nodes p o s i t i o n

Nn= l e n g t h ( V e c t o r \_ n o d e s ) ; % t o t a l number of nodes Ne=Nn−1; % t o t a l number of e l e m e n t s V e c t o r \_ e l = ( 1 : 1 : Ne ) ; % e l e m e n t n u m e r o t a i o n

% e l e m e n t e r y s t i f n e s s m a t r i x Ke=ES \*[ 1 −1;−1 1 ] / h ;

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

Notre méthode de décompositions de domaine consiste à fixer d’abord la taille *h* (ligne 7), puis à poser un nombre d’élé- ment par sous-domaine *ne* (ligne 8). Cela nous amène pour une longueur fixe *L* (ligne 5) au nombre des sous-domaines *N ss* (ligne 10) et à la taille *H* (ligne 9). La matrice de rigidité élémentaire est donnée par la ligne 17 .

Tout d’abord, on cherche la matrice de rigidité globale par sous-domaine c’est l’objectif de la boucle for (ligne 4 ). Dans notre cas, la matrice est la même pour tous les sous-domaines. Par conséquent il suffirait de la calculer une seule fois. Ensuite, on réécrit cette matrice sous la forme (10), (lignes 16-24). Le calcul des paramètres de Schur (expression 13

) suit (lignes 26-34 ). En introduisant l’opérateur d’assemblage *A* qui permet de récupérer les ddl de déplacement aux bords, 39-44, et après l’élimination du degré de liberté correspondant à l’encastrement, on obtient les paramètres de Schur primaux *Sp* et *Bp* assemblés (lignes 49 et 50).

1 %% s t i f n e s s m a t r i x

2 Ks= z e r o s ( Nn , Nn ) ; % " s " d e s i g n a t e s t h e s u b s t r u c t u r e

3 % s u b s t r u c t u r e ’ s s t i f n e s s m a t r i x

4 f o r s = 1 : round ( Nss )

5 f o r j = 1 : Ne

6 Ks ( j , j + 1 , s ) =Ke ( 1 , 2 ) ;

7 Ks ( j + 1 , j , s ) =Ke ( 1 , 2 ) ;

8 Ks ( 1 , 1 , s ) =Ke ( 1 , 1 ) ;

9 Ks ( end , end , s ) =Ke ( 1 , 1 ) ;

10 Ks ( j , j , s ) =2\*Ke ( 1 , 1 ) ;

11 end

12 end

13 % Schr p a r a m e t e r s

14 f o r s = 1 : round ( Nss )

15 % b bord / i i n t e r n

16 Ks\_bb ( : , : , s ) = [ Ks ( 1 , 1 ) Ks ( 1 , end ) ; Ks ( end , 1 ) Ks ( end , end ) ] ;

17 Ks\_ib ( : , : , s ) = [ Ks ( 2 : Ne , 1 ) Ks ( 2 : Ne , end ) ] ;

18 Ks\_bi ( : , : , s ) = Ks\_ib ( : , : , s ) ’ ;

19

20 f o r i = 2 : Ne

21 f o r j = 2 : Ne

22 K s \_ i i ( i −1 , j −1 , s ) = Ks ( i , j , s ) ;

23 end

24 end

25 %c a l c u l de Sp

26 Sp\_s ( : , : , s ) = Ks\_bb ( : , : , s ) −( Ks\_bi ( : , : , s ) \*( i n v ( K s \_ i i ( : , : , s ) ) ) \* Ks\_ib ( : , : , s ) ) ;

27 f i ( : , 1 , s ) = z e r o s ( Ne− 1 , 1 ) ;

28 fb ( [ 1 , 2 ] , 1 , s ) = z e r o s ( 2 , 1 ) ;

29 fb ( [ 1 , 2 ] , 1 , Nss ) = [ 0 ; Fd ( 1 , 1 ) ] ;

30 bp\_s ( : , 1 , s ) = fb ( [ 1 , 2 ] , 1 , s ) −( Ks\_bi ( : , : , s ) \*( i n v ( K s \_ i i ( : , : , s ) ) ) \* f i ( : , 1 , s ) ) ;

31 x =( s −1)\*2+1 ;

32 y= ( s −1) \* 2 + 2 ;

33 Bp ( [ x y ] , 1 ) =bp\_s ( : , 1 , s ) ; % Schur 2 nd member <| >

34 Sp ( [ x y ] , [ x y ] ) =Sp\_s ( : , : , s ) ; % Schur p r i m a l compl <\ >

35 a ( s , 1 , s ) = 1 ; % o p e r a t o r A f o r s u b s t r u c t u r e s

36 a ( s + 1 , 2 , s ) = 1 ;

37 end

38 % o p e r a t o r A <−>

39 f o r s = 1 : round ( Nss )

40 x = ( s −1)\*2 + 1 ;

41 y = ( s −1)\*2 + 2 ;

42 A ( : , x ) =a ( : , 1 , s ) ;

43 A ( : , y ) =a ( : , 2 , s ) ;

44 end

45 % r e d u c e d Schur P a r a m e t e r s

46 Sp ( dof\_blocked , : ) = [ ] ; Sp ( : , d o f \_ b l o c k e d ) = [ ] ;

47 Bp ( dof\_blocked , : ) = [ ] ;

48 A( dof\_blocked , : ) = [ ] ; A ( : , d o f \_ b l o c k e d ) = [ ] ;

49 SP = A\* Sp\*A’ ;

50 BP = A\*Bp ;

La résolution directe du problème donne exactement la bonne solution (Figure 6). Ceci étant, pour résoudre le système, on peut utiliser une méthode itérative, plus efficace dans le cas général. C’est l’objectif de la section 3.2.3 .

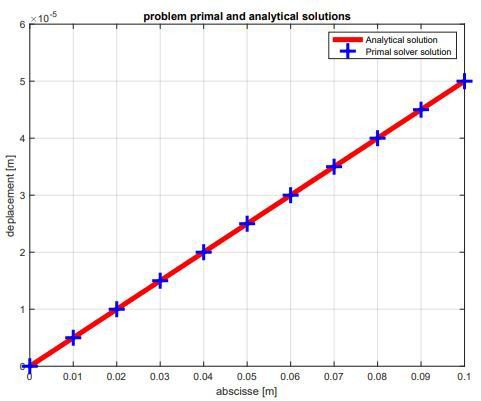


FIGURE 6 – Déplacement selon x, solution analytique et solution via une méthode de Schur primal assemblée.

### Conditionnement de la matrice de Schur primal et de K

On étudie les variations du conditionnement de Sp et de K en fonction de *h/H* et de *h*. On utilise pour cela la fonction cond sur MATLAB (ligne 23 pour K et ligne 56 pour Sp) . Pour différents nombre de sous-domaines *Ns* [1-10] (boucle for ligne2 ) et pour différents nombre d’éléments par sous-domaine *ne* [1-20] (boucle for ligne3), on obtient les tracés ci-dessous, figures 7 et 8.

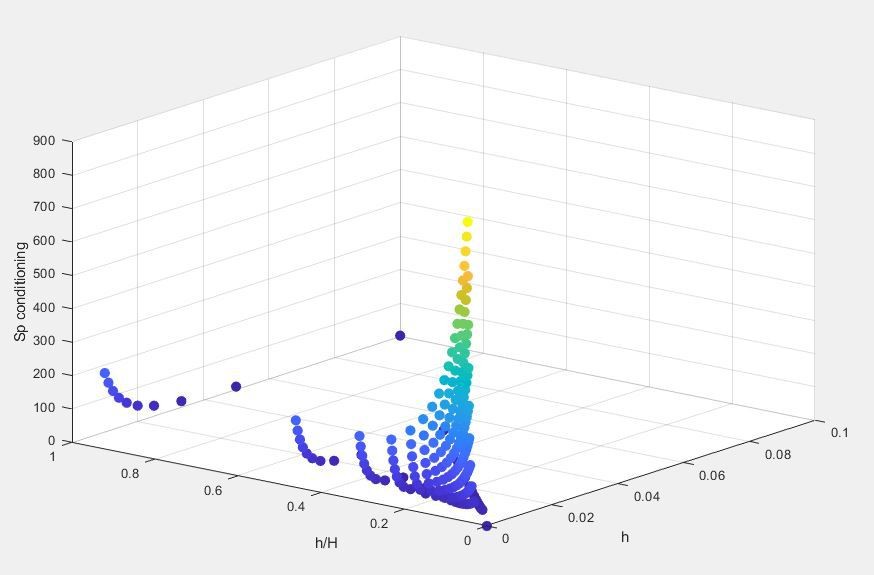


FIGURE 7 – Conditionnement de Sp en norme 2.

Discussion :

on observe lorsque le rapport *h/H* diminue, une augmentation importante du conditionnement de *Sp*. De plus, une diminution de *h* génère une augmentation du conditionnement. Il faut donc trouver une position de compromis. Le système sera d’autant plus facile à résoudre que l’on a des grands éléments et des sous-domaines à quelques éléments. Si l’on met trop d’éléments dans un sous domaine, le système à inversé est mal conditionné, et si l’on met des sous-domaines trop petits, on perd l’intérêt de la décomposition de domaine. Un choix raisonnable de *h* et *H* dépendra en pratique du cas à traiter et des méthodes de parallélisation utilisées.

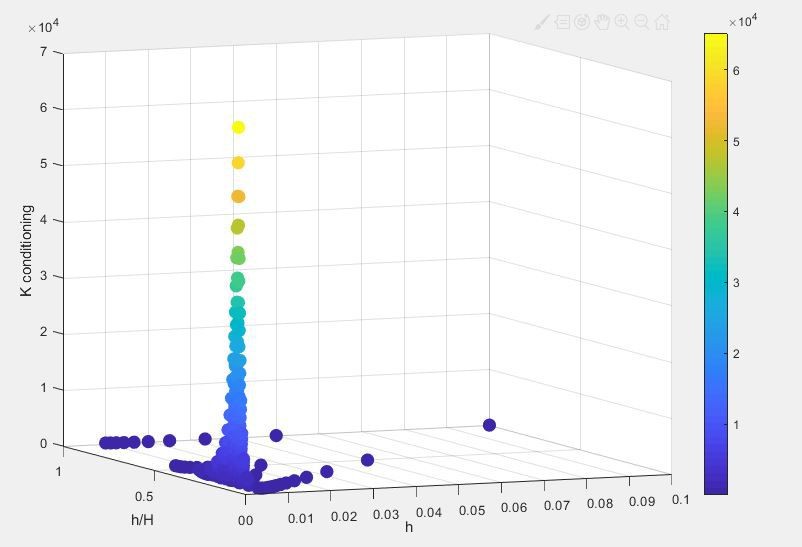


FIGURE 8 – Conditionnement de K en norme 2

Discussion :

pour le conditionnement de K on remarque que plus la taille des éléments *h* est petite, plus le conditionnement de *K* est grand. En effet, plus on augmente le nombre d’éléments par sous-domaines, plus on se rapproche d’une résolution éléments finis classique avec quelque sous-domaines, et nécessairement, l’augmentation de taille de *K* entraîne une aug- mentation de son conditionnement. Il faut donc limiter le nombre d’éléments par sous-domaine.

f o r

Ns = 1 : 10

f o r ne = 1 : 20

Ne = ne \*Ns ; h = L / Ne ;

H = L / Ns ;

% t o t a l Number of e l e m e n t s

v e c t o r \_ h ( Ns , ne ) =h ;

v e c t o r \_ h \_ p e r \_ H ( Ns , ne ) =h / H; Ke = ES \*[ 1 −1; −1 1 ] / h ;

Ks= z e r o s ( Ne+ 1 , Ne+ 1 ) ; K= z e r o s ( Ne+ 1 , Ne+ 1 ) ; K s \_ i i = z e r o s ( Ne−1) ; Ks\_bi= z e r o s ( 2 , Ne−1) ; Ks\_ib= z e r o s ( Ne− 1 , 2 ) ; Bs\_bb= z e r o s ( 2 ) ;

% G l o b a l s t i f n e s m a t r i x

f o r j = 1 : Ne

K( j , j + 1 ) =Ke ( 1 , 2 ) ;

K( j + 1 , j ) =Ke ( 1 , 2 ) ;

K( 1 , 1 ) =Ke ( 1 , 1 ) ;

K( end , end ) =Ke ( 1 , 1 ) ;

K( j , j ) =2\*Ke ( 1 , 1 ) ;

end

%BC

K ( : , 1 ) = [ ] ; K ( 1 , : ) = [ ] ;

c o n d i t i o n i n g K ( Ns , ne ) =cond (K) ;

%Sp c o n d i t i o n n i n g K\_s = z e r o s ( Ne+ 1 ) ; f o r i = 1 : Ne

K\_s ( i , i ) =K\_s ( i , i ) +Ke ( 1 , 1 ) ;

K\_s ( i + 1 , i ) =K\_s ( i + 1 , i ) +Ke ( 2 , 1 ) ;

K\_s ( i , i + 1 ) =K\_s ( i , i + 1 ) +Ke ( 1 , 2 ) ;

K\_s ( i + 1 , i + 1 ) =K\_s ( i + 1 , i + 1 ) +Ke ( 2 , 2 ) ;

end

% Decomposition of K x = K\_s ( 1 , : ) ;

K\_s ( 1 : Ne − 1 , : ) = K\_s ( 2 : Ne , : ) ;

K\_s ( Ne , : ) = x ;

x = K\_s ( : , 1 ) ;

K\_s ( : , 1 : Ne−1) = K\_s ( : , 2 : Ne ) ;

K\_s ( : , Ne ) = x ;

% Complement Schur p r i m a l

Ks\_bb = K\_s ( Ne : Ne+ 1 , Ne : Ne+ 1 ) ; K s \_ i i = K\_s ( 1 : Ne− 1 , 1 : Ne−1) ;

Ks\_ib = K\_s ( 1 : Ne−1 ,Ne : Ne+ 1 ) ; Ks\_bi = K\_s ( Ne : Ne + 1 , 1 : Ne−1) ;

S\_p\_s = Ks\_bb−Ks\_bi \* i n v ( K s \_ i i ) \* Ks\_ib ;

% assembled o p e r a t o r s

S\_p= z e r o s ( Ns + 1 ) ; f o r i = 1 : Ns

S\_p ( i , i ) =S\_p ( i , i ) +S\_p\_s ( 1 , 1 ) ;

S\_p ( i + 1 , i ) =S\_p ( i + 1 , i ) +S\_p\_s ( 2 , 1 ) ;

S\_p ( i , i + 1 ) =S\_p ( i , i + 1 ) +S\_p\_s ( 1 , 2 ) ;

S\_p ( i + 1 , i + 1 ) =S\_p ( i + 1 , i + 1 ) +S\_p\_s ( 2 , 2 ) ;

end

%BC’ s

S\_p ( 1 , : ) = [ ] ;

S\_p ( 1 , 1 ) =( ES / h ) ;

c o n d i t i o n n e m e n t S ( Ns , Ne ) = cond ( S\_p ) ;

end

end

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

32

33

34

35

36

37

38

39

40

41

42

43

44

45

46

47

48

49

50

51

52

53

54

55

56

57

58

### Résolution du problème par la méthode du gradient conjugué parallélisé

Ici, la méthode n’est pas numériquement extensible. En faisant varier le ratio *h/H* ou encore les tailles *H* et *h*, on aboutit à différent nombre d’itération et temps de calcul. La solution obtenue pour les déplacement aux interface est visible sur la figure 9. La méthode converge correctement et la solution obtenue est correcte.

10-5 **problem primal and analytical solutions**

6

Analytical solution

Primal conjugated parallelized solver solution

5

4

3

deplacement [m]

2

1

0

0 0.01 0.02 0.03 0.04 0.05 0.06 0.07 0.08 0.09 0.1

abscisse [m]

FIGURE 9 – Solution obtenue via un gradient conjugué distribué.

1 ub\_bold = z e r o s ( N b \_ i n t e r f a c e , 1 ) ;

2 ub\_assembled = A\_assembled ’\* ub\_bold ;

3 f o r i = 1 : Nb\_sub

4 ub\_sub ( : , : , i ) = ub\_assembled ( 2 \* ( i −1) + 1 : 2 \* i ) ;

5 end

6 % c a l c u l du r sidu i n i t i a l

7 f o r i = 1 : Nb\_sub

8 r b \_ s u b ( : , : , i ) = −( Sp\_sub \* ub\_sub ( : , : , i )−bp\_sub ( : , : , i ) ) ;

9 end

10 r b \_ a s s e m b l e d = r b \_ s u b ( : , : , 1 ) ;

11 f o r i = 2 : Nb\_sub

12 r b \_ a s s e m b l e d = [ r b \_ a s s e m b l e d ; r b \_ s u b ( : , : , i ) ] ;

13 end

14 r \_ b o l d = A\_assembled \* r b \_ a s s e m b l e d ;

15 % d

16 d\_bold = r \_ b o l d ;

17 % c r i t re d ’ a r r t

18 e p s i l o n = r\_bold ’\* r \_ b o l d ;

19 % compteur d ’ i t ration

20 compteur = 0 ;

21 % b o u c l e

22 w h i l e e p s i l o n > 10 e−3

23 % compteur d ’ i t ration

24 compteur = compteur + 1 ;

25 % c a l c u l du pas o p t i m a l

26 d\_assembled = A\_assembled ’ \* d\_bold ;

27 f o r i = 1 : Nb\_sub

28 d\_sub ( : , : , i ) = d\_assembled ( 2 \* ( i −1) + 1 : 2 \* i ) ;

29 end

30 f o r i = 1 : Nb\_sub

31 v\_sub ( : , : , i ) = Sp\_sub \* d\_sub ( : , : , i ) ;

32 end

33 v\_assembled = v\_sub ( : , : , 1 ) ;

34 f o r i = 2 : Nb\_sub

35 v\_assembled = [ v\_assembled ; v\_sub ( : , : , i ) ] ;

36 end

37 v\_bold = A\_assembled \* v\_assembled ;

38 a l p h a = ( r\_bold ’\* d\_bold ) / ( d\_bold ’\* v\_bold ) ;

39 % c a l c u l de l ’ i t r ub s t e p +1

40 ub\_bold = ub\_bold + a l p h a \* d\_bold ;

41 ub\_assembled = A\_assembled ’\* ub\_bold ;

42 % d composition p a r sous−s t r u c t u r e

43 f o r i = 1 : Nb\_sub

44 ub\_sub ( : , : , i ) = ub\_assembled ( 2 \* ( i −1) + 1 : 2 \* i ) ;

45 end

46 % c a l c u l du nouveau r sidu

47 f o r i = 1 : Nb\_sub

48 r b \_ s u b ( : , : , i ) = −( Sp\_sub \* ub\_sub ( : , : , i )− bp\_sub ( : , : , i ) ) ;

49 end

50 r b \_ a s s e m b l e d = r b \_ s u b ( : , : , 1 ) ;

51 f o r i = 2 : Nb\_sub

52 r b \_ a s s e m b l e d = [ r b \_ a s s e m b l e d ; r b \_ s u b ( : , : , i ) ] ;

53 end

54 r b \_ b o l d \_ o l d = r \_ b o l d ;

55 r \_ b o l d = A\_assembled \* r b \_ a s s e m b l e d ;

56 % c a l c u l du t erme d ’ o r t h o g o n a l i s a t i o n

57 b e t a = ( r\_bold ’\* r \_ b o l d ) / ( r b \_ b o l d \_ o l d ’\* r b \_ b o l d \_ o l d ) ;

58 % mise j o u r d i r e c t i o n de r e c h e r c h e

59 d\_bold = r \_ b o l d + b e t a \* d\_bold ;

60 % mise j o u r c r i t re d ’ a r r t

61 e p s i l o n = r\_bold ’\* r \_ b o l d ;

62 end

### Résolution du problème par la méthode du gradient conjugué pré-conditionné.

Ici, on ne peut pas conclure sur l’extensibilité numérique, car la solution est obtenue dès l’étape de pré-conditionnement. Ainsi, aucune itération n’est nécessaire, que ce soit pour une discrétisation fine ou grossière du maillage. La solution obtenue pour les déplacement aux interface est visible sur la figure 10, on obtient bien une solution correcte.

10-5 **problem primal and analytical solutions**

6

Analytical solution

Primal preconditionned solver solution

5

4

3

deplacement [m]

2

1

0

0 0.01 0.02 0.03 0.04 0.05 0.06 0.07 0.08 0.09 0.1

abscisse [m]

FIGURE 10 – Solution obtenue via un gradient conjugué pré-conditionné

On s’intéresse au pseudo-conditionnement du système. Ici, on utilise un pseudo-inverse du complément de Schur primal au lieu de son inverse. Le conditionnement d’une matrice s’écrit en norme 2 :

*cond*(*A*) = ||*A−*1||2 ||*A*||2*.* (14)

Dans notre cas, en utilisant la norme de la matrice pseudo inverse, on trouve des conditionnement qui sont toujours de l’ordre de 1 pour différents ratio h/H et valeur de H. On a donc un système très bien conditionné.

1 A\_assembled = A\_assembled ( 2 : end , : ) ;

2 Sp\_bold = A\_assembled \* Sp\_assembled \* A\_assembled ’ ;

3 bp\_bold = A\_assembled \* bp\_assembled ;

4 % d finition de M comme l a m a t r i c e i d e n t i t

5 M = eye ( s i z e ( A\_assembled , 2 ) ) ;

6 % d finition de A t i l d e

7 A \_ t i l d e = i n v ( A\_assembled \*M\* A\_assembled ’ ) \* A\_assembled \*M;

8 % d fintition de Sp ^1 t i l d e

9 S p \_ t i l d e \_ i n v = A \_ t i l d e \* p i nv ( Sp\_assembled ) \* A \_ t i l d e ’ ;

10 % d finition de G t i l d e

11 G \_ t i l d e = A \_ t i l d e \* Rb\_assembled ;

12 % d fintion du p r o j e c t e u r P

13 P = eye ( s i z e ( G \_ t i l d e , 1 ) ) − G \_ t i l d e \* i n v ( G \_ t i l d e ’\* Sp\_bold \* G \_ t i l d e ) \* G \_ t i l d e ’\* Sp\_bold

14 % c a l c u l du d placement i n i t i a l , s u p p o s n u l

15 ub\_bold = G \_ t i l d e \* i n v ( G \_ t i l d e ’\* Sp\_bold \* G \_ t i l d e ) \* G \_ t i l d e ’\* bp\_bold ;

16 % c a l c u l du r sidu i n i t i a l

17 r \_ b o l d = P ’\* bp\_bold ;

18 z \_ b o l d = S p \_ t i l d e \_ i n v \* r \_ b o l d ;

19 % d

20 d\_bold = z \_ b o l d ;

21 % c r i t re d ’ a r r t

22 e p s i l o n = r\_bold ’\* r \_ b o l d ;

23 % compteur d ’ i t ration

24 compteur = 0 ;

25 % b o u c l e

26 w h i l e e p s i l o n > 10 e−3

27 % compteur d ’ i t ration

28 compteur = compteur + 1 ;

29 % c a l c u l du pas o p t i m a l

30 p\_bold = P ’\* Sp\_bold \* d\_bold ;

31 a l p h a = ( r\_bold ’\* d\_bold ) / ( d\_bold ’\* p\_bold ) ;

32 % c a l c u l de l ’ i t r ub s t e p +1

33 ub\_bold = ub\_bold + a l p h a \* d\_bold ;

34 % d composition p a r sous−s t r u c t u r e

35 f o r i = 1 : Nb\_sub

36 ub\_sub ( : , : , i ) = ub\_assembled ( 2 \* ( i −1) + 1 : 2 \* i ) ;

37 end

38 % c a l c u l du nouveau r sidu

39 r b \_ b o l d \_ o l d = r \_ b o l d ;

40 r \_ b o l d = r b \_ b o l d \_ o l d − a l p h a \* p\_bold ;

41 z \_ b o l d = S p \_ t i l d e \_ i n v \* r \_ b o l d ;

42 % c a l c u l du t erme d ’ o r t h o g o n a l i s a t i o n

43 f o r j = 1 : compteur

44 b e t a ( j ) = −( z\_bold ’\* p\_bold ) / ( d\_bold\_old ’\* p\_bold ) ;

45 end

46 somme = b e t a ( 1 )

47 i f compteur >= 2

48 f o r j = 2 : compteur

49 somme = somme + b e t a ( j ) ;

50 end

51 end

52 % mise j o u r d i r e c t i o n de r e c h e r c h e

53 d\_bold = z \_ b o l d + somme\* d\_bold ;

54 % mise j o u r c r i t re d ’ a r r t

55 e p s i l o n = r\_bold ’\* r \_ b o l d ;

56 end

# Résolution par décomposition de domaine – approche duale

## l’approche duale

Dans l’approche duale, on utilise un unique multiplicateur de Lagrange *λ* aux interfaces afin de prendre en compte la contrainte de continuité des déplacements. Le problème est résolu en déterminant la valeur de ces multiplicateurs, et la solution est alors connue à un mouvement de corps rigide près. Le système à résoudre devient le système 15

(*s*)



( ) = A *K*(*s*)

(*s*)

*K*

*ib*

B A *u*(*s*)

B = A *f* (*s*)

B + A *λ*(*s*) = 0 B

∀*s* ∈ ***S****, K u*

*s* *ii*

*K*



(*s*)

*bi*

(*s*) *.*

*bb*

*K*

(*s*)

*b*

*i*

*u*

(*s*)

*b*

*i*

*f*

(*s*)

*b*

*i*

*λ*

*R*(*s*)*T* (*f*



(*s*) + *λ*(*s*)) = 0

(15)

 ∀Γ(*s,s'*)*,* (*s, s'*) (*s*) + *λ*(*s'*) = 0

∈ ***S****, λb*

*b*

Afin de résoudre ce problème, il faut dans un premier temps déterminer les modes de corps rigide. Ceux ci s’appliquent uniquement aux sous-domaines flottants, c’est à dire sans condition imposé de déplacement. Dans notre cas, seul le sous- domaine 1 est non-flottants. Le problème est 1D donc seul le déplacement axial est pris en compte comme mode rigide. La matrice *R*(*s*) est de taille *nnodes,s* x taille du vecteur des modes rigides, ici de taille 1. On a donc simplement des vecteurs colonnes remplis de 1 pour tous les sous-domaines, sauf le premier.

## résolution du système dual assemblé