

DEA Physique et Génie des Matériaux

—

Mastère Matériaux et Mise en Forme

—

TD Élément Finis - **Solutions**

P. Laure (Patrice.Laure@inln.cnrs.fr)  
Institut Non Linéaire de Nice



## Correction 1 : Initiation à Scilab

**Ex 1** *Manipulation sur les vecteurs*

1)

```
v1 = 1:.1:3
```

2)

```
v2 = 3:-.1:1
```

3)

```
v3 = (1:10)^2
```

4)

```
n = (1:10);  
un = ones(1,10);  
v4 = (- un).^n .* (n^2)
```

5)

```
un = ones(1,10)  
v5 = [0*un, un]
```

**Ex 2** *Manipulation sur les Matrices*

1)

```
m = matrix(1:36,6,6)'  
m = [1:6;7:12;13:18;19:24;25:30;31:36]
```

2)

```
n = 5  
C = zeros(n,n);  
for i = 1:n  
    C(i,i) =2;  
end;  
for i = 1:n-1  
    C(i,i+1) = -1;  
end;  
for i = 2:n  
    C(i,i-1) = -1;  
end;
```

ou dans un style plus Scilab,

```
C = 2*eye(n,n)-diag(ones(n-1,1),1)-diag(ones(n-1,1),-1)}
```

### Ex 3 Exemple de tracé

```
X = 0:.1:4*pi;  
Y = sin(X) + X;  
plot2d(X,Y)  
xtitle('Exemple','x','sin(x) + x');
```

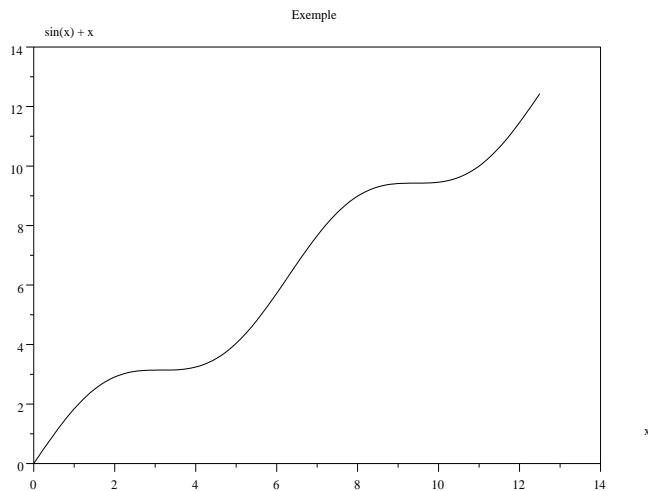


FIG. 1 – Tracé de  $f(x) = \sin x + x$ .

### Ex 4 Ecriture d'une fonction

```
function C = alterne2_colonne(A,B)
```

```
[n,m] = size(A)
```

```
C = []
```

```
for i=1:m
```

```
    C = [C, A(:,i), B(:,i)];
```

```
end;
```

ou avec un style plus **scilab**

```
function C = alterne2_colonne(A,B)
```

```
[n,m] = size(A)
```

```
A = matrix(A,n*m,1);
```

```
B = matrix(B,n*m,1);
```

```
C = [A B]
```

```
C = matrix(C,n,2*m);
```

```
endfunction
```

**Ex 5** *Calcul de la solution d'une edo*

1)

C'est une équation à coefficients constants : une solution particulière de l'équation est  $u_s = 1$ ; la solution générale de l'équation sans second membre  $u_0 = a e^x + b e^{-x}$ , alors la solution peut s'écrire  $u = u_s + u_0$ .

Les conditions aux limites permettent de calculer les coefficients  $a$  et  $b$  ...

2)

```
clear
```

```
n = 50;
```

```
F = ones(n,1);
```

```
A = eye(n,n);
```

```
x = matrix(linspace(0,1,n),n,1);
```

```
Dx = 1/(n-1);
```

```
u_exact = 1- (exp(x) + exp(1-x))/(1+exp(1));
```

```
// on impose les conditions aux limites
```

```
F(1) = 0;
```

```
F(n) = 0;
```

```
for i=2:n-1
```

```
    A(i,i) = A(i,i) + 2./Dx^2;
```

```
    A(i,i-1) = -1/Dx^2;
```

```
    A(i,i+1) = -1/Dx^2;
```

```
end;
```

```
u_c = A\F;
```

```
erreur = norm(u_exact-u_c)
```

```
\\ Trace de la solution
```

```
xbasc()
```

```
plot2d(x,u_c,style=[-1])
```

```
x=linspace(0,1,100);
```

```
u_exact = 1- (exp(x) + exp(1-x))/(1+exp(1));
```

```
plot2d(x,u_exact,style=[1])
```

```
xtitle('Solution','x','u(x)');
```

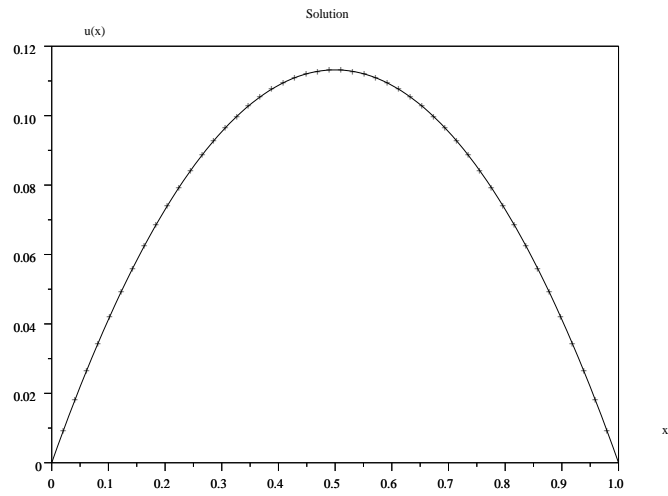


FIG. 2 – Solution du problème pour  $n = 50$  : solution exacte (-), solution calculée (+).

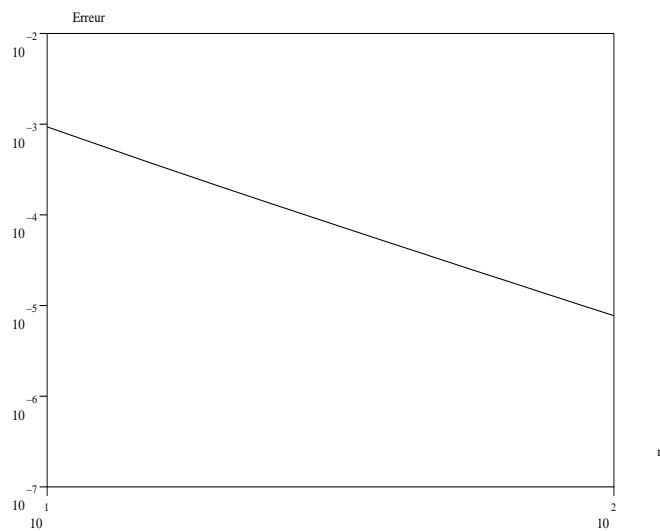


FIG. 3 – Evolution de l'erreur  $\|u - u_{exact}\|/\|u_{exact}\|$  en fonction de  $n$

## Correction 2 : Initiation aux éléments finis 1D

Ex 1 *Éléments Finis Unidimensionnels*

1)

La formulation faible

$$\int_0^1 (u' v' + u v) dx = \int_0^1 v dx$$

2)

Le listing complet du programme est donné à la réponse de la question 5).

Pour le calcul de COOR et de CONNEC, il faut lire les lignes 8-24.

3)

Le système élémentaire sur le  $k$ ème élément s'écrit :

$$A^k U^k = F^k \quad (1)$$

où :

$$a_{ij}^k = \int_{x_1^k}^{x_2^k} [\psi_j^{k'} \psi_i^{k'} + \psi_j^k \psi_i^k] dx = \int_{-1}^1 \left[ \left( \frac{1}{Jac^k} \right)^2 \frac{dL_j}{d\xi} \frac{dL_i}{d\xi} + L_j L_i \right] Jac^k d\xi$$

$$f_i^k = \int_{x_1^k}^{x_2^k} \psi_i^k dx = \int_{-1}^1 L_i Jac^k d\xi$$

où  $Jac^k$  est le Jacobien de la transformation

$$Jac^k = \frac{x_2^k - x_1^k}{2}$$

c.f. lignes 30-50 du listing.

4)

c.f. lignes 55-62 du listing.

5)

Listing du programme permettant la résolution avec l'élément  $P_1$  :

```

1 // on charge les programmes dont on a besoin
2 //
3 exec('P1.sci'); exec('gauss.sci');
4 // nbre de points pour l'integration numerique
5 Ng = 3;
6 [XG,PG] = gauss(Ng);
7 //
8 //
9 Nelt = 20; // nbre d'elements
10
11 Nno = 2; // nbre de noeud par element

```

```

12
13 Npt = Nelt * (Nno-1)+1; // nbre de points
14
15 Dx = 1./(Npt-1);
16
17 COOR = matrix((0:Dx:1),Npt,1); // Coordonnees des points
18
19 CONNEC = zeros(Nelt,Nno);
20 for i=1:Nelt
21     for j=1:Nno
22         CONNEC(i,j) = j + (i-1); // matrice de connectivite
23     end;
24 end;
25 //
26 // Fin initialisation des tableaux
27
28 A_glob = zeros(Npt,Npt); F_glob = zeros(Npt,1);
29
30 for iel = 1:Nelt
31     // calcul du systeme elementaire sur chaque element
32     A_el = zeros(Nno,Nno);
33     F_el = zeros(Nno,1);
34
35     x1 = COOR(CONNEC(iel,1)); x2 = COOR(CONNEC(iel,2));
36     jac = (x2 - x1)/2.; // le jacobien est constant sur l'element
37
38     for ig=1:Ng // boucle pour l'integration numerique
39
40         [FP,DxFP] = P1(XG(ig)); // choix des fonctions tests
41
42         DxFP = DxFP/jac;
43
44         for i=1:Nno
45             F_el(i) = F_el(i) + PG(ig)*FP(i)*jac;
46             for j=1:Nno
47                 A_el(i,j) = A_el(i,j) + PG(ig)*(DxFP(j)*DxFP(i) + FP(i)*FP(j))*jac;
48             end;
49         end;
50     end;
51 // fin calcul systeme elementaire pour l'element iel
52
53 // on doit assembler
54
55 for i=1:Nno
56     i_glob = CONNEC(iel,i);
57     F_glob(i_glob) = F_glob(i_glob) + F_el(i);
58     for j=1:Nno
59         j_glob = CONNEC(iel,j);
60         A_glob(i_glob,j_glob) = A_glob(i_glob, j_glob) + A_el(i,j);
61     end;
62 end;

```



```

63 end; // fin boucle sur les elements
64
65 //
66 // On impose les deux conditions aux limites de Diriclet
67 //      U(0) = U(1) = 0
68
69 A = [zeros(1,Npt); A_glob(2:(Npt-1),:); zeros(1,Npt)];
70 A(1,1) = 1; A(Npt,Npt) = 1;
71 F = [0; F_glob(2:(Npt-1)); 0];
72
73 u_cal = A\F;
74
75 // comparaison avec la solution exacte
76 //
77 x = COOR;
78 u_exact = 1- (exp(x) + exp(1-x))/(1+exp(1));
79 erreur = norm(u_cal-u_exact)/norm(u_exact)
80
81 xbasec();
82 plot2d(x,u_cal,style=-1);
83 x = linspace(0,1,100);
84 u_exact = 1- (exp(x) + exp(1-x))/(1+exp(1));
85 plot2d(x,u_exact,style=1);
86 xtitle('Solution P1; Nelt =' + string(Nelt) + ', Npt =' + string(Npt), 'x', 'u');
87

```

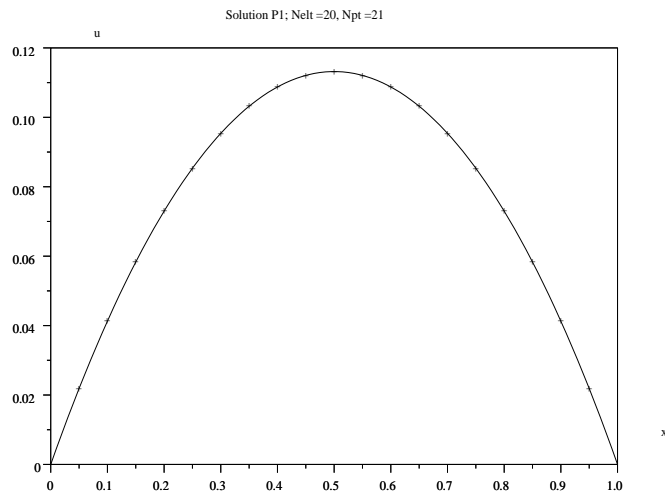


FIG. 4 – Solution du problème pour  $N_{elt} = 20$  : solution exacte (—), solution calculée (+).

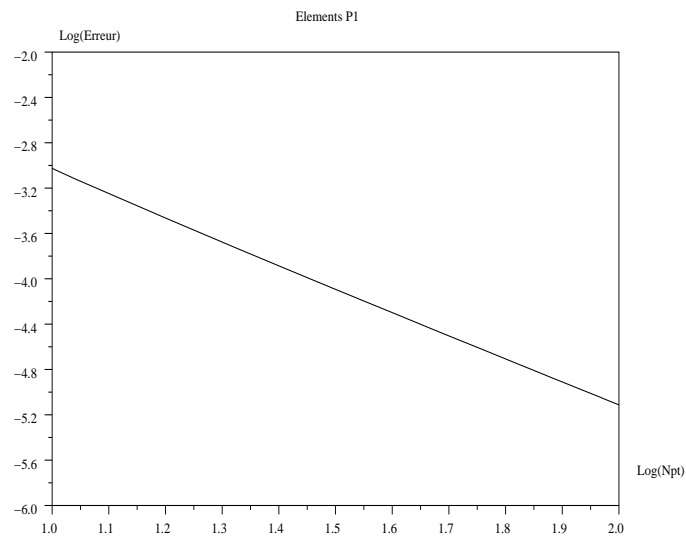


FIG. 5 – Evolution de l'erreur  $\|u - u_{exact}\|/\|u_{exact}\|$  en fonction de  $N_{pt}$

6)

Si on veut utiliser des éléments  $\mathbf{P}_2$ , les modifications sont mineures par rapport au programme précédent :

- Il faut appeler la fonction qui calcule les polynômes de Lagrange de degré 2 (Ligne 3).
- Modifier la valeur `Nno` (Ligne 11).
- Le calcul de la matrice de connectivité (Lignes 20-24).
- L'appel aux fonctions tests  $L_2$  lors de l'intégration numérique (Ligne 40).

```
3   exec('P2.sci'); exec('gauss.sci');

11  Nno = 3; // nbre de noeuds par element

20  for i=1:Nelt
21      CONNEC(i,1) = 1 + 2*(i-1);
22      CONNEC(i,2) = 3 + 2*(i-1);
23      CONNEC(i,3) = 2 + 2*(i-1);
24  end;

40  [FP,DxFP] = P2(XG(ig));
```

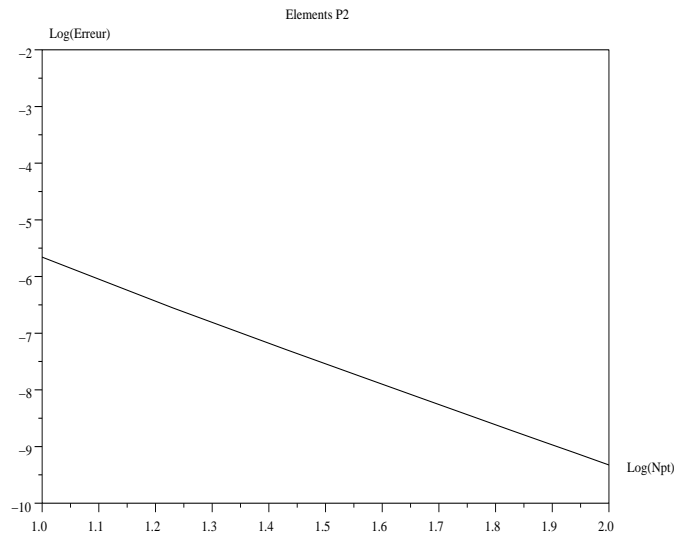


FIG. 6 – Evolution de l'erreur  $\|u - u_{exact}\|/\|u_{exact}\|$  en fonction de  $\mathbf{Npt}$  pour des éléments  $\mathbf{P}_2$ .

7)

les modifications sont les suivantes :

On libère la valeur de  $u$  en  $x = 1$  aux lignes 69-71.

```
69  A = [zeros(1,Npt); A_glob(2:Npt,:)];
70  A(1,1) = 1;
71  F = [0; F_glob(2:Npt)];
```

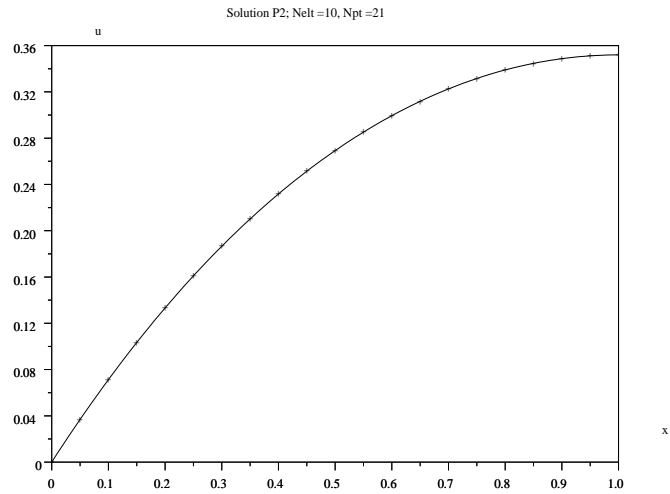


FIG. 7 – Solution du problème pour  $N_{el} = 10$  avec des éléments  $\mathbf{P}_2$  : solution exacte (–), solution calculée (+).

## Correction 3 : Initiation aux éléments finis 2D

Ex 1 *Écoulements de Poiseuille*

1)

La formulation faible

$$\int_0^1 \int_0^1 \left( \frac{\partial u}{\partial x} \frac{\partial v}{\partial x} + \frac{\partial u}{\partial y} \frac{\partial v}{\partial y} \right) dx dy = \int_0^1 \int_0^1 f v dx dy + \int_0^1 \left[ \frac{\partial u}{\partial x} v \right]_{x=0}^{x=1} dy + \int_0^1 \left[ \frac{\partial u}{\partial y} v \right]_{y=0}^{y=1} dx$$

Sur chaque élément, on obtient le système élémentaire  $\mathbf{N} \mathbf{u} = \mathbf{F}$  suivant

$$A^k U^k = F^k \quad (2)$$

où :

$$a_{ij}^k = \int \int_{\mathbf{K}} [\nabla \psi_j^k]^T [\nabla \psi_i^k] dx dy \quad \text{avec} \quad [\nabla \psi_j^k] = \begin{pmatrix} \frac{\partial \psi_j^k}{\partial x} \\ \frac{\partial \psi_j^k}{\partial y} \end{pmatrix}$$

$$f_i^k = \int \int_{\mathbf{K}} \psi_i^k dx dy$$

2)

La matrice Jacobienne

$$DT_k = \begin{pmatrix} \sum_{j=1}^4 x_j^k \frac{\partial L_j}{\partial \xi} & \sum_{j=1}^4 x_j^k \frac{\partial L_j}{\partial \eta} \\ \sum_{j=1}^4 y_j^k \frac{\partial L_j}{\partial \xi} & \sum_{j=1}^4 y_j^k \frac{\partial L_j}{\partial \eta} \end{pmatrix}$$

Ce qui donne le programme

```

x = COOR(CONNEC(iel,:),1); y = COOR(CONNEC(iel,:),2);

[F,DxiFP,DetaFP] = Q1(xg,yg); // on evalue les fonctions de Lagrange au point (xg,yg)

j11 = x(1:4)'*DxiFP; // = sum_j x_j^k dL_j/dxi
j12 = x(1:4)'*DetaFP;
j21 = y(1:4)'*DxiFP;
j22 = y(1:4)'*DetaFP;
//
DT = [j11 j12; j21 j22];

```

3)

Listing du programme

```
1  clear; xdel(0); xdel(1);
2  // on charge les programmes dont on a besoin
3  //
4  exec('Q1.sci'); exec('Q2.sci'); exec('gauss.sci');
5  exec('maillage_2D_Q2.sci');
6  exec('cal_indice.sci'); exec('interpol.sci');
7  exec('trace_maillage_2D.sci'); exec('trace_u_2D.sci');
8
9  //
10 // nbre de points pour l'integration numerique
11 Ng = 3;
12 [XG,PG] = gauss(Ng);
13 //
14 nx = 10; ny = 10; // nbre d'element dans les directions x et y
15 lx = 1.; ly = 1.; // taille du carre
16
17 [CONNEC,COOR,Nelt,Nno,Npt] = maillage_2D_Q2(1,1,nx,ny);
18 trace_maillage_2D(CONNEC,COOR,0);
19
20 //
21 // creation de la matrice NUMER et ADRESS
22 //
23 // Conditions aux limites
24 // u = 0 sur tout le bord
25 NUMER=zeros(Npt,1);
26 NI = 0;
27 nc = 0;
28 for i=1:Npt
29     x = COOR(i,1); y = COOR(i,2);
30     if x <> 0 & x <> 1 & y <> 0 & y <> 1 then
31         NI = NI+1;
32         NUMER(i) = NI;
33     else
34         NUMER(i) = Npt-nc;
35         nc = nc + 1;
36     end;
37 end;
38 //
39 // creation de la matrice ADRESS
40 //
41 ADRESS = zeros(Nelt,Nno);
42 for i =1:Nelt
43     for j = 1:Nno
44         ADRESS(i,j) = NUMER(CONNEC(i,j));
45     end;
46 end;
```

```

47 //
48 // creation du vecteur UC
49 //
50 UC = zeros(Npt-NI,1);
51
52 //
53 // Fin initialisation des tableaux
54
55 A_glob = spzeros(Npt,Npt); F_glob = zeros(Npt,1);
56
57 for iel = 1:Nelt // boucle sur les elements
58 //
59 // -----
60 // PARTIE A ECRIRE
61 //-----
62 // calcul du sytème elementaire sur chaque element
63 //
64 A_el = zeros(Nno,Nno);
65 F_el = zeros(Nno,1);
66
67 x = COOR(CONNEC(iel,:),1); y = COOR(CONNEC(iel,:),2);
68
69
70 for ig=1:Ng // double boucle pour l'integration numerique
71 xg = XG(ig);
72 for jg=1:Ng
73 yg = XG(jg);
74 //
75 // calcul de la matrice jacobienne
76 //
77 [F,DxiFP,DetaFP] = Q1(xg,yg);
78 j11 = x(1:4)'*DxiFP;
79 j12 = x(1:4)'*DetaFP;
80 j21 = y(1:4)'*DxiFP;
81 j22 = y(1:4)'*DetaFP;
82 //
83 DT = [j11 j12; j21 j22];
84 jac = det(DT);
85 B = inv(DT)';
86 //
87 [FP,DxiFP,DetaFP] = Q2(xg,yg);
88
89 for i=1:Nno
90 DFPi = B*[DxiFP(i); DetaFP(i)];
91 F_el(i) = F_el(i) + PG(ig)*PG(jg)*FP(i)*jac;
92 for j=1:Nno
93 DFPj = B*[DxiFP(j); DetaFP(j)];
94 A_el(i,j) = A_el(i,j) + PG(ig)*PG(jg)*( DFPj' * DFPi ) *jac;
95 end;
96 end;
97 end;

```

```

98     end;
99     //
100    // fin calcul systeme elementaire pour l'element iel
101    //
102    //-----
103    //  FIN PARTIE A ECRIRE
104    //-----
105    // on doit assembler
106
107    for i=1:Nno
108        i_glob = ADRESS(iel,i);
109        F_glob(i_glob) = F_glob(i_glob) + F_el(i);
110    end;
111
112    A_el=sparse(A_el);
113    [ij,v,mn] = spget(A_el);
114    for i = 1:length(v)
115        ij(i,1) = ADRESS(iel,ij(i,1));
116        ij(i,2) = ADRESS(iel,ij(i,2));
117    end;
118    A_el = sparse(ij,v,[Npt Npt]);
119    A_glob = A_glob + A_el;
120
121 end; // fin boucle sur les elements
122
123
124 A = A_glob(1:NI,1:NI); F = F_glob(1:NI);
125 M12 = A_glob(1:NI,NI+1:$);
126 F = F - M12*UC;
127
128 [h,rk] = lufact(A);
129 U = lusolve(h,F);
130 ludel(h);
131
132 u_cal = [U; UC];
133
134 [u,x,y]=trace_u_2D(u_cal,COOR,NUMER,1,20,20);

```



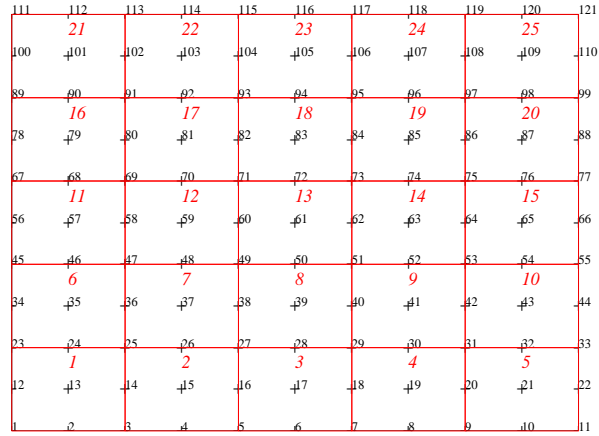


FIG. 8 – Exemple de maillage avec  $N_{elt} = 25$  et  $N_{pt} = 121$ .

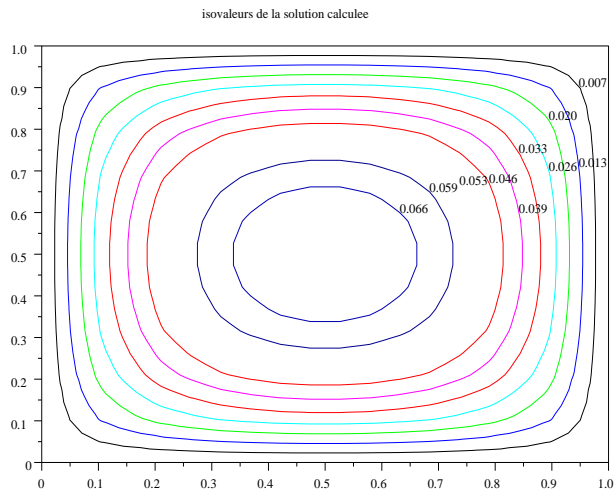


FIG. 9 – Solution du problème pour avec  $N_{elt} = 100$  et  $N_{pt} = 441$ .

4)

La formule qui donne le débit est :

$$q = \int_0^1 \int_0^1 u(x, y) dx dy = \sum_{k=1}^{N_{elt}} \left( \sum_{j=1}^{N_{no}} u_j^k \int_{-1}^1 L_j(\xi, \eta) Jac^k d\xi d\eta \right)$$

avec  $u|_{K_k} = \sum_{j=1}^{N_{no}} u_j^k \psi_j^k(x, y)$ .

On trouve  $q = .0351$  avec le programme suivant :

```
function q = cal_debit_2D(U,COOR, CONNEC, NUMER, ADRESS)
//
[Nelt, Nno] = size(CONNEC);
Npt = length(COOR);

q = 0.;
for iel = 1:Nelt

    x = COOR(CONNEC(iel,:),1); y = COOR(CONNEC(iel,:),2);
    u_el = U(ADRESS(iel,:))

    for ig=1:Ng // double boucle pour l'integration numerique
        xg = XG(ig);
        for jg=1:Ng
            yg = XG(jg);
//
// calcul de la matrice jacobienne
//
            [F,DxiFP,DetaFP] = Q1(xg,yg);
            j11 = x(1:4)'*DxiFP;
            j12 = x(1:4)'*DetaFP;
            j21 = y(1:4)'*DxiFP;
            j22 = y(1:4)'*DetaFP;
//
            DT = [j11 j12; j21 j22];
            jac = det(DT);
//
            [FP,DxiFP,DetaFP] = Q2(xg,yg);
//
            for i=1:Nno
                q = q + PG(ig)*PG(jg) * u_el(i)*FP(i) * jac;
            end;
        end;
    end;
end; // fin boucle sur les elements

endfunction
```

5)

Il faut changer le calcul de la matrice NUMER à la ligne 28. Il faut “libérer” les degrés de liberté qui sont situés sur les bords  $x = 0$  et  $1$ . on remplace la condition

```
if x <> 0 & x <> 1 & y <> 0 & y <> 1 then
```

par

```
if y <> 0 & y <> 1 then
```

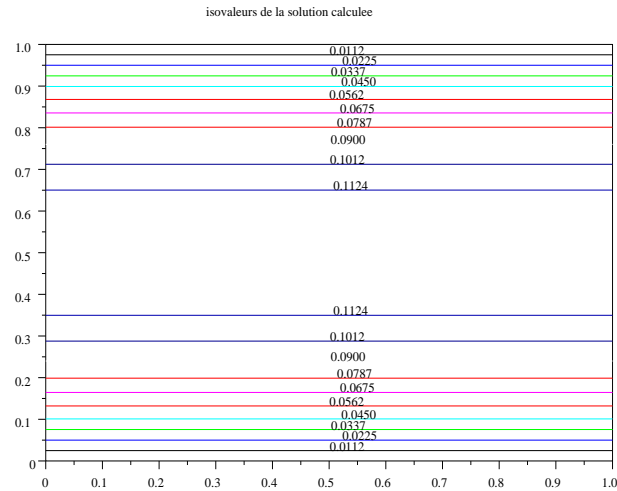


FIG. 10 – Solution de la question 5) pour avec  $N_{elt} = 100$  et  $N_{pt} = 441$ .

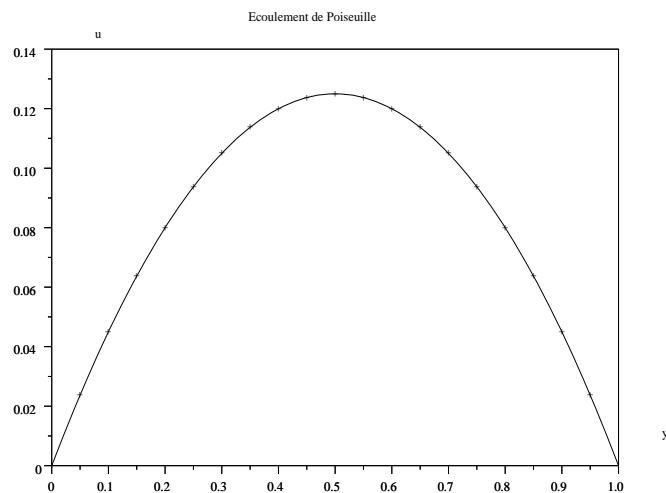


FIG. 11 – Comparaison entre la solution calculée (+) et la solution analytique  $u_e = y(y - 1)/2$  pour une valeur de  $x$  fixée.

**Ex 2** *Ecoulement analytique*

1)

La formulation faible

$$\int_0^1 \int_0^1 \left( \frac{\partial u}{\partial x} \frac{\partial v}{\partial x} + \frac{\partial u}{\partial y} \frac{\partial v}{\partial y} \right) dx dy + 3 \int_0^1 \int_0^1 u v dx dy = 0$$

2)

Le programme est essentiellement le même que pour l'exercice précédent. Il suffit d'imposer les conditions aux limites de Diriclet correctement

```
UC = zeros(Npt-NI,1);
for i=1:Npt
  x = COOR(i,1); y = COOR(i,2);
  if y == 0 | y == 1 | x == 0 | x == 1 then
    UC(NUMER(i)-NI) = exp(2*x) * sin(y);
  end;
end;
```

et de calculer correctement la matrice élémentaire

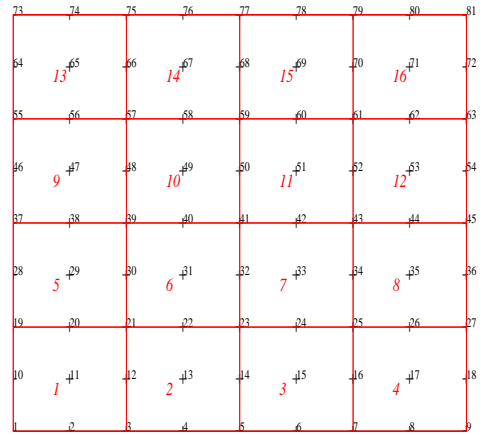
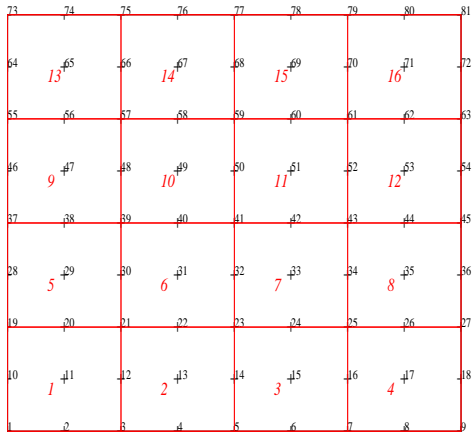
```
A_el(i,j) = A_el(i,j) + ...
  PG(ig)*PG(jg)*( DFPj(1)*DFPi(1) + DFPj(2)*DFPi(2) + 3*FP(j)*FP(i) ) *jac;
```

Si on utilise des éléments  $\mathbf{P}_1$ , il faut utiliser la quadrature de Hammer et la fonction `hammer.sci` pour avoir les points d'intégration.

Quelques exemples de calcul sont rassemblés dans le tableau suivant :

	Npt	Nelt	Nno	$\ u_c - u_a\ /\ u_a\ $
$\mathbf{Q}_1$	81	64	4	$5.05 \cdot 10^{-4}$
$\mathbf{Q}_2$	81	16	9	$1.40 \cdot 10^{-5}$
$\mathbf{P}_1$	81	128	3	$5.15 \cdot 10^{-4}$

TAB. 1 – Erreur relative  $\|u_c - u_a\|/\|u_a\|$  en fonction des éléments choisis.



U

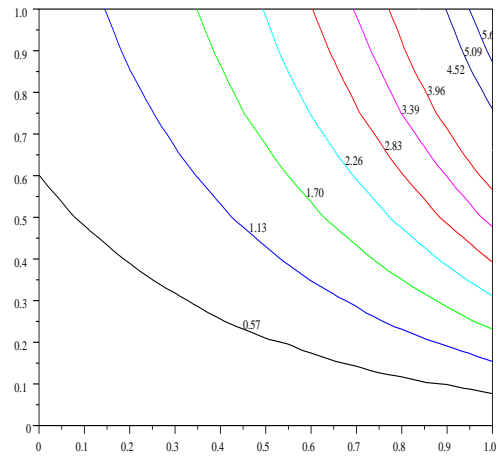
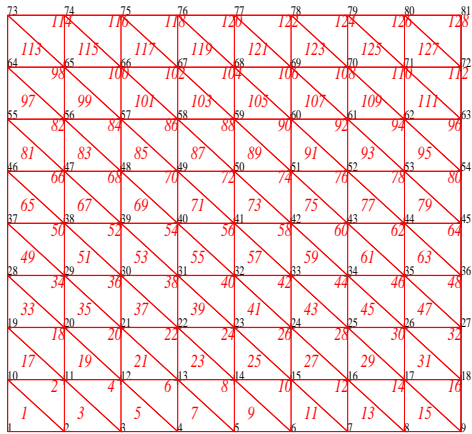


FIG. 12 – Les différents maillages et la solution : (a)  $\mathbf{Q}_1$ ; (b)  $\mathbf{Q}_2$ ; (c)  $\mathbf{P}_1$ ; (d)  $u(x, y)$ .



---

**Correction 4 : Programmation modulaire - Problème non linéaires**
**Ex 1** *Écoulement de Poiseuille*

1)

En 1D l'équation s'écrit

$$\frac{\partial}{\partial x} \left( \mu \frac{\partial u}{\partial x} \right) = -\nabla P$$

La formulation faible

$$\int \mu \frac{\partial u}{\partial x} \frac{\partial w}{\partial x} dx = \nabla P \int w dx \quad (3)$$

**3.1)**Si on utilise une méthode du point fixe, pour calculer  $u^{n+1}$  à partir de  $u^n$ , on doit résoudre

$$\int_0^1 \mu(u^n) \frac{\partial u^{n+1}}{\partial x} \frac{\partial w}{\partial x} dx = \nabla P \int_0^1 w dx$$

ce qui donne pour le calcul de la matrice élémentaire le programme suivant

```
function [A_el,F_el] = cal_Melm_1D_a(iel,gradP,U);
//
    global CONNEC COOR NUMER ADRESS UC Nelt Nno Npt NI;
    global XG PG Ng;
//
    x = COOR(CONNEC(iel,:),1)

    A_el = zeros(Nno,Nno); F_el = zeros(Nno,1);

    U_el= U(ADRESS(iel,:));

    jac = (x(2) - x(1))/2.;

    for ig=1:Ng // boucle pour l'integration numerique
        xg = XG(ig);
//
        [FP,DxiFP] = P2(xg);
        DxFP = DxiFP/jac;

        gp = abs(U_el'*DxFP);
        mu = cal_mu(gp);

        for i=1:Nno
            F_el(i) = F_el(i) + PG(ig)*gradP*FP(i)*jac;
            for j=1:Nno
                A_el(i,j) = A_el(i,j) + mu * PG(ig) * DxFP(j) * DxFP(i) * jac;
            end;
        end;
    end;
```

```

end;
endfunction

```

### 3.2)

La méthode du point fixe correspond aux instructions suivantes

```

// on calcule la solution par une methode de pt fixe
//
function U = cal_sol_ptfixe(cal_Melm,gradP,U)
//
//
    eps = .0001; h = .01; imax = 50;

    U0 = cal_sol_lin(cal_Melm,gradP,U);

    i = 0;
    while norm(U-U0) >= eps & i <= imax
        i = i + 1;
        U0 = U;
        U = cal_sol_lin(cal_Melm,gradP,U0);

        trace_u_1D(U,1);
        fprintf(%io(2),'i = %3i ; ||U-U0|| = %f',i,norm(U-U0));

    end;

endfunction

```

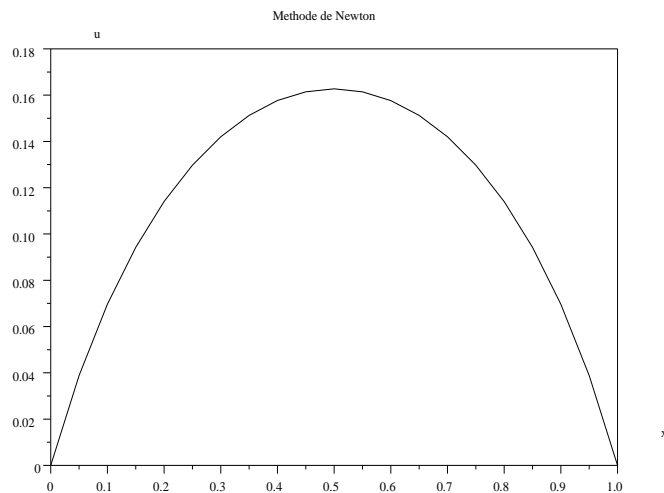


FIG. 13 – Ecoulement de Poiseuille pour  $\mu_0 = 1.$ ,  $\lambda = 2.$ ,  $m = .2$  et  $a = 2.$

### 3.3)

Si on utilise une méthode de Newton,  $u^{n+1} = u^n + \delta u$ , où  $u^n$  a été calculé à l'étape  $n$  et  $\delta u$  est solution



de l'équation (3) "linéarisée",

$$\int_0^1 \left( \mu(u^n) \frac{\partial(\delta u)}{\partial x} \frac{\partial w}{\partial x} + \mu'(u^n) \operatorname{sgn}(\partial_x u^n) \frac{\partial(\delta u)}{\partial x} \frac{\partial u^n}{\partial x} \frac{\partial w}{\partial x} \right) dx$$

$$= \nabla P \int w dx - \int_0^1 \mu(u^n) \frac{\partial u^n}{\partial x} \frac{\partial w}{\partial x} dx$$
(4)

avec

$$\mu' = \mu \frac{(m-1)(\lambda \dot{\epsilon})^a}{\dot{\epsilon}(1+(\lambda \dot{\epsilon})^a)}$$
(5)

ce qui donne pour le calcul du système élémentaire le programme suivant

```
function [A_el,F_el] = cal_Melm_1D_b(iel,gradP,U);
//
global CONNEC COOR NUMER ADRESS UC Nelt Nno Npt NI;
global XG PG Ng;
//
zero = 0.001;
x = COOR(CONNEC(iel,:),1)

A_el = zeros(Nno,Nno); F_el = zeros(Nno,1);

U_el= U(ADRESS(iel,:));

jac = (x(2) - x(1))/2.;

for ig=1:Ng // boucle pour l'integration numerique
    xg = XG(ig);
//
    [FP,DxiFP] = P2(xg);
    DxFP = DxiFP/jac;

    gradU = U_el'*DxFP;
    gp = abs(gradU);

    mu = cal_mu(gp);
    mup = sign(gradU)*cal_Dmu(gp);

    for i=1:Nno
        F_el(i) = F_el(i) + PG(ig)*( gradP*FP(i) ...
            - mu * gradU * DxFP(i) ) *jac;
;
        for j=1:Nno
            A_el(i,j) = A_el(i,j) + ...
                PG(ig)*( mu * DxFP(j) * DxFP(i) + ...
                    mup*DxFP(i)* gradU *DxFP(j) ) * jac;
        end;
    end;
end;
endfunction
```

Pour la méthode de Newton, on modifie légèrement le programme du points fixe :

```
// on calcule la solution par une methode de Newton
//
function U = cal_sol_newton(cal_Melm,gradP,U)
//
    eps = .0001; h = .01; imax = 50;

    du = cal_sol_lin(cal_Melm,gradP,U);

    i = 0;
    while norm(du) >= eps & i <= imax
        i = i + 1;
        du = cal_sol_lin(cal_Melm,gradP,U);
        U = U + du;

        trace_u_1D(U,2);
        fprintf(%io(2),'i = %3i ; ||du|| = %f',i,norm(du));
    end;

endfunction
```

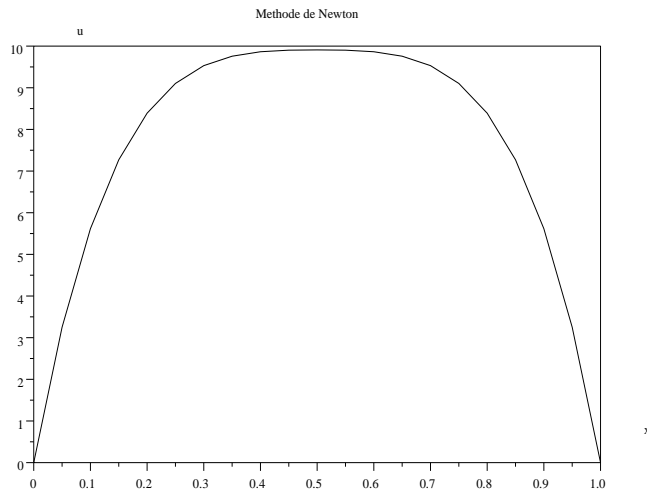


FIG. 14 – Ecoulement de Poiseuille pour  $\mu_0 = 1.$ ,  $\lambda = 13.47$ ,  $m = 0.294$  et  $a = .381$ .

### 3.2)

On peut remarquer que la méthode de Newton converge plus vite que la méthode du point fixe. En particulier la méthode du point fixe n'arrive pas à calculer l'écoulement de Poiseuille pour le polyéthylène.

Les resultats peuvent se resumer dans le tableau suivant :

### 4)

<i>paramètres</i>	<i>débit</i>	<i>Iter. Mth. pt. Fixe</i>	<i>iter. Mth. Newton</i>
$\mu_0 = 1., \lambda = 2., m = .2$ et $a = 2$	0.114	13	5
$\mu_0 = 1., \lambda = 13.47, m = 0.294$ et $a = .381$	7.795	33	8

TAB. 2 – Comparaison du nombre d’itération pour atteindre la solution.

On écrit l’équation sous forme développée :

$$\frac{\partial}{\partial x} \left( \mu \frac{\partial u}{\partial x} \right) + \frac{\partial}{\partial y} \left( \mu \frac{\partial u}{\partial y} \right) = -\nabla P$$

On a la formulation faible

$$\begin{aligned} \int \mu \left[ \frac{\partial u}{\partial x} \frac{\partial w}{\partial x} + \frac{\partial u}{\partial y} \frac{\partial w}{\partial y} \right] d\Omega &= \nabla P \int w d\Omega + \\ &+ \int_0^1 \mu \left[ \frac{\partial u}{\partial y}(x, 1)w(x, 1) - \frac{\partial u}{\partial y}(x, 0)w(x, 0) \right] dx \\ &+ \int_0^1 \mu \left[ \frac{\partial u}{\partial x}(x, 1)w(1, y) - \frac{\partial u}{\partial x}(x, 0)w(0, y) \right] dy \end{aligned}$$

avec les conditions aux limites, on a  $w(0, y) = w(1, y) = w(x, 0) = w(x, 1) = 0$  pour la base des fonctions tests.

si  $\mu$  n’est pas constant on peut trouver une solution avec des méthodes itératives.

#### La méthode de point fixe :

Si à l’étape  $n$  on connaît  $u^n$ , la solution  $u^{n+1}$  à l’étape suivante est donnée par

$$\int \mu(u^n) \left[ \frac{\partial u^{n+1}}{\partial x} \frac{\partial w}{\partial x} + \frac{\partial u^{n+1}}{\partial y} \frac{\partial w}{\partial y} \right] d\Omega = \nabla P \int w d\Omega$$

#### La méthode de Newton :

Si à l’étape  $n$  on connaît  $u^n$ , on note  $u^{n+1} = u^n + \delta u$  la solution à l’étape suivante où  $\delta u$  est solution de l’équation linéarisée

$$\begin{aligned} \int \left( \mu(u^n) \left[ \frac{\partial \delta u}{\partial x} \frac{\partial w}{\partial x} + \frac{\partial \delta u}{\partial y} \frac{\partial w}{\partial y} \right] + \frac{\mu'(u^n)}{\bar{\epsilon}(u^n)} \left[ \frac{\partial \delta u}{\partial x} \frac{\partial u^n}{\partial x} + \frac{\partial \delta u}{\partial y} \frac{\partial u^n}{\partial y} \right] \left[ \frac{\partial u^n}{\partial x} \frac{\partial w}{\partial x} + \frac{\partial u^n}{\partial y} \frac{\partial w}{\partial y} \right] \right) d\Omega \\ = \nabla P \int w d\Omega - \int \mu(u^n) \left[ \frac{\partial u^{n+1}}{\partial x} \frac{\partial w}{\partial x} + \frac{\partial u^{n+1}}{\partial y} \frac{\partial w}{\partial y} \right] d\Omega \end{aligned}$$

<i>paramètres</i>	<i>débit</i>	<i>Iter. Mth. pt. Fixe</i>	<i>iter. Mth. Newton</i>
$\mu_0 = 1., \lambda = 2., m = .2$ et $a = 2$	0.0385	6	4
$\mu_0 = 1., \lambda = 13.47, m = 0.294$ et $a = .381$	1.2826	28	7

TAB. 3 – Comparaison du nombre d’itération pour atteindre la solution avec `Nel1t` = 100.

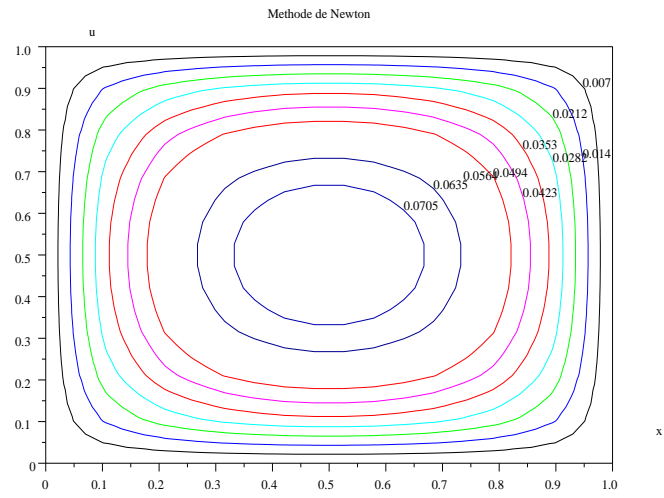


FIG. 15 – Ecoulement de Poiseuille pour  $\mu_0 = 1.$ ,  $\lambda = 2.$ ,  $m = .2$  et  $a = 2.$

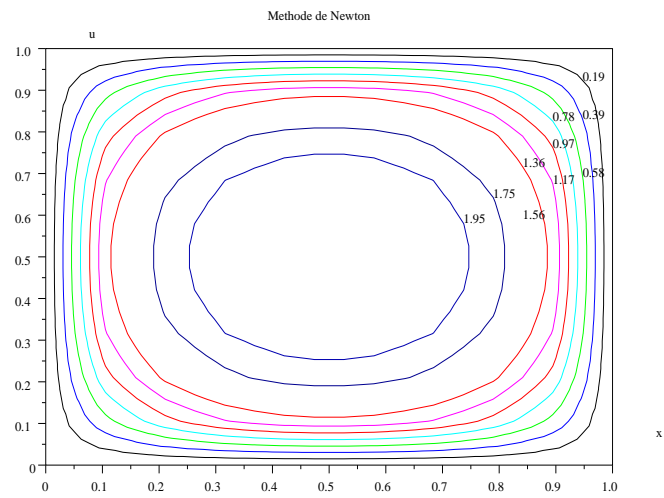


FIG. 16 – Ecoulement de Poiseuille pour  $\mu_0 = 1.$ ,  $\lambda = 13.47$ ,  $m = 0.294$  et  $a = .381.$



**Correction 5 : Eléments finis mixtes**

On veut résoudre le problème de Stokes

$$\nabla \cdot \sigma + f = 0$$

avec  $\sigma = 2\mu(\nabla u + (\nabla u)^T) - pE$  et  $\nabla \cdot u = 0$ . Sous forme développée

$$\sigma = 2\mu \begin{pmatrix} \frac{\partial u}{\partial x} & \frac{1}{2} \left( \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right) \\ \frac{1}{2} \left( \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right) & \frac{\partial v}{\partial x} \end{pmatrix}$$

On écrit l'équation sous forme développée :

$$\begin{aligned} \frac{\partial}{\partial x} \left( 2\mu \frac{\partial u}{\partial x} \right) + \frac{\partial}{\partial y} \left( \mu \left( \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right) \right) - \frac{\partial p}{\partial x} + f_x &= 0 \\ \frac{\partial}{\partial x} \left( \mu \left( \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right) \right) + \frac{\partial}{\partial y} \left( 2\mu \frac{\partial v}{\partial y} \right) - \frac{\partial p}{\partial y} + f_y &= 0 \\ \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} &= 0 \end{aligned}$$

On a la formulation variationnelle si  $\phi = (w_1, w_2, q)$  est la fonction test

$$\begin{aligned} \int_{\Omega} \left[ 2\mu \frac{\partial u}{\partial x} \frac{\partial w_1}{\partial x} + \mu \left( \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right) \frac{\partial w_1}{\partial y} - p \frac{\partial w_1}{\partial x} - f_x w_1 \right] dx dy - \int_{\Gamma} \tau_x w_1 ds &= 0 \\ \int_{\Omega} \left[ \mu \left( \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right) \frac{\partial w_2}{\partial x} + 2\mu \frac{\partial v}{\partial y} \frac{\partial w_2}{\partial y} - p \frac{\partial w_2}{\partial x} - f_y w_2 \right] dx dy - \int_{\Gamma} \tau_y w_2 ds &= 0 \\ \int_{\Omega} \left[ \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \right] q dx dy &= 0 \end{aligned}$$

avec

$$\begin{aligned} \tau_x &= \left( 2\mu \frac{\partial u}{\partial x} - p \right) n_x + \mu \left( \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right) n_y \\ \tau_y &= \left( \mu \left( \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right) n_x + 2\mu \frac{\partial v}{\partial x} - p \right) n_y \end{aligned}$$

Si on considère un carré avec les conditions aux limites

$$\begin{aligned} u(x, 0) = v(x, 0) = v(x, 1) = 0 \quad \text{et} \quad u(x, 1) = 1 \\ u(0, y) = v(0, y) = u(1, y) = v(1, y) = 0 \end{aligned}$$

$\tau_x$  et  $\tau_y$  n'interviennent pas car on prend des fonctions tests qui s'annulent sur les cotés.

Sur chaque élément, on utilise des fonctions d'interpolation différentes pour la vitesse ( $N_j^u(x, y)$ ) et la pression ( $N_j^p(x, y)$ )

$$u = \sum_{j=1}^n u_j N_j^u \quad ; \quad v = \sum_{j=1}^n v_j N_j^v \quad ; \quad p = \sum_{j=1}^m p_j N_j^p$$

Sur l'élément, on obtient le système local

$$\begin{bmatrix} K^{11} & K^{12} & K^{13} \\ K^{21} & K^{22} & K^{23} \\ K^{31} & K^{32} & 0 \end{bmatrix} \begin{bmatrix} \{u\} \\ \{v\} \\ \{p\} \end{bmatrix} = \begin{bmatrix} F^1 \\ F^2 \\ 0 \end{bmatrix}$$

avec

$$K_{ij}^{11} = \int_{\Omega} \mu \left( 2 \frac{\partial N_j^u}{\partial x} \frac{\partial N_i^u}{\partial x} + \frac{\partial N_j^u}{\partial y} \frac{\partial N_i^u}{\partial y} \right) dx dy ; \quad K_{ij}^{12} = \int_{\Omega} \mu \frac{\partial N_j^u}{\partial x} \frac{\partial N_i^u}{\partial y} dx dy ; \quad K_{ij}^{13} = - \int_{\Omega} N_j^p \frac{\partial N_i^u}{\partial x} dx dy$$

$$K_{ij}^{21} = \frac{\partial N_j^u}{\partial y} \frac{\partial N_i^u}{\partial y} ; \quad K_{ij}^{22} = \int_{\Omega} \mu \left( \frac{\partial N_j^u}{\partial x} \frac{\partial N_i^u}{\partial x} + 2 \frac{\partial N_j^u}{\partial y} \frac{\partial N_i^u}{\partial y} \right) dx dy ; \quad K_{ij}^{23} = - \int_{\Omega} N_j^p \frac{\partial N_i^u}{\partial y} dx dy$$

$$K_{ij}^{31} = \int_{\Omega} \frac{\partial N_j^u}{\partial x} N_i^p dx dy ; \quad K_{ij}^{32} = \int_{\Omega} \frac{\partial N_j^u}{\partial y} N_i^p dx dy$$