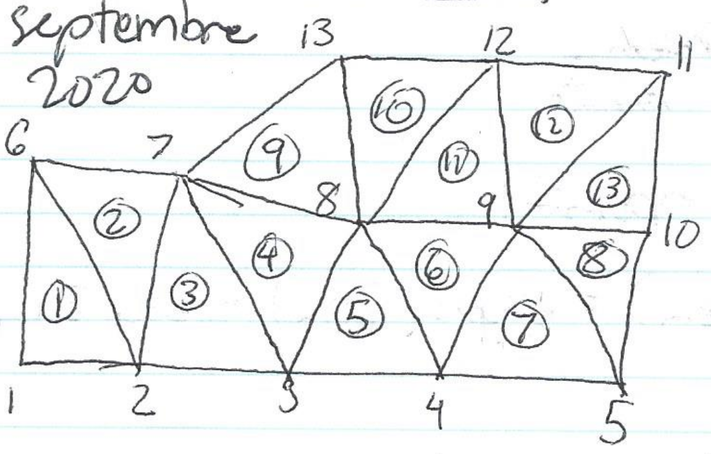
Maillage

22 septembre 2020



nelem = 13
nnode = 3

npoin = 13 (hasard)

Inpoel

	élem #1	élem #2	3	4	5	6	7	8	9	10	11	12	13
nd #1	1	2	2	3	3	4	4	5	7	8	8	9	9
nd #2	2	7	3	8	4	9	5	10	8	12	9	11	10
nd #3	6	6	7	7	8	8	9	9	13	13	12	12	11

init. esup2 (1 : npoin + 1) = 0 => [000 00000000000]
14 x 0

```
do ielem=1, nelem
  do inode=1, nnode
      ipoil = inpoel (inode, ielem) +1
      esup2 (ipoil) = esup2 (ipoil) +1
  enddo
enddo
```

ex : ielem = 1 et inode = 1 => ipoil = 2
=> esup2(2) = 1

inode = 2 => ipoil = 3 esup2(3) = 1
inode = 3 => ipoil = 7 esup2(7) = 1

[0 1 1 0 0 0 1 0 0 0 0 0 0 0]

O.L

~~do ielem=1, nelem~~
~~do inode=1, nnode~~

```
do  ipoin = 2, npoin + 1
      esup2(ipoin) = esup2(ipoin) + esup2(ipoin-1)
enddo
```

Element Pass 1 : Count the number of elements connected to each point

ielem = 2 inode = 1 ipoi = 3 esup2(3) = 1+1 = 2

inode = 2 ipoi = 8 esup2(8) = 1

inode = 3 ipoi = 7 esup2(7) = 2

~~[0 1 1 0 0 0]~~

1 2 3 4 5 6 7 8 9 10 11 12 13 14

2 iterations [0 1 2 0 0 0 2 1 0 0 0 0 0 0]

à la fin de premier de ⇒ esup2 = [0 1 3 3 3 2 2 4 6 6 2 2 3 2]

je do (Storage / reshuffling pass 1) :

esup2 = [0 1 4 7 10 12 14 18 24 30 32 34 37 39

esup2 est terminé !!!

33d

esup2 = [0 1 4 7 10 12 14 18 24 30 32 34 37 39]

Element Pass 2 : Store the elements in esup1

```
do ielem=1, nelem
   do inode=1, nnode
      ipoin = inpoel(inode, ielem)
      istor = esup2(ipoin)+1
      esup2(ipoin) = istor
      esup1(istor) = ielem
   enddo
enddo
```

ielem=1, inode=1 => ipoin = 1 (→ 70)
istor = esup2(1)+1 = 1
esup2(1) = 1
esup1(1) = 1 [1 1 4 ...]

inode=2 => ipoin = 2
istor = 1+1 = 2
esup2(2) = 2
esup1(2) = 1 [1 2 4 4
[1 1

inode 3 => ipoin = 6
istor = 12+1

esup1 = [1 1 0 0 0 0 0 0 0 0 0 0 1]

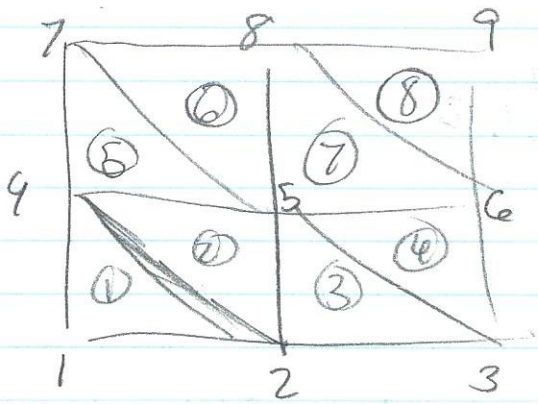
esup1 = [

ielem = 2, inode = 1 => ipoint = 2
istar = 3

[ ]

esup1 => "elements surrounding point"

Ok!



NNODE = 3
NPOIN = 9
NELEM = 8

esup1 = [1 ; 1 2 3 ; 3 4 ; 1 2 5 ; 2 3 4 5 6 7 ; 4 7 8 ; 5 6 ; 6 7 8 ; 8]

longueur = NELEM * NNODE

ipoin = 0
istar = 1
esup2(0) = 1
esup1(0) = 0+1

ipoin = 2-1 = 1
istar = 1+1 = 2
esup2(1) = 2
esup1(1) = 0+1

ipoin = 6-1 = 5
istar = 12+1
esup2(5) = 12

i=0

ipoin ?
istar - 1

23 septembre 2020

++i plus i++

```
for (size_t i=0; i<5; ++i)
```

%.2 affiche centièmes
%.2f float

Tableaux : taille N => indices 0 -> N-1

```
double A1[12];
int A2[2] = {1, 2};
```

```
M1[5][4]
```

```
int main(void)
```

ne retourne rien
type entier

```
#include <math.h>
#include <vector>
```

push_back

lire et écrire dans un fichier header <fstream>

psup1 = [2 6 | 1 3 6 7 | 2 4 7 8 | 3 5 8 9 |
4 9 10 | 1 2 7 | 2 3 6 8 13 |
3 4 7 9 12 13 | 4 5 8 10 11 12 |
5 9 11 | 9 10 12 | 8 9 11 13 | 7 8 12]

50

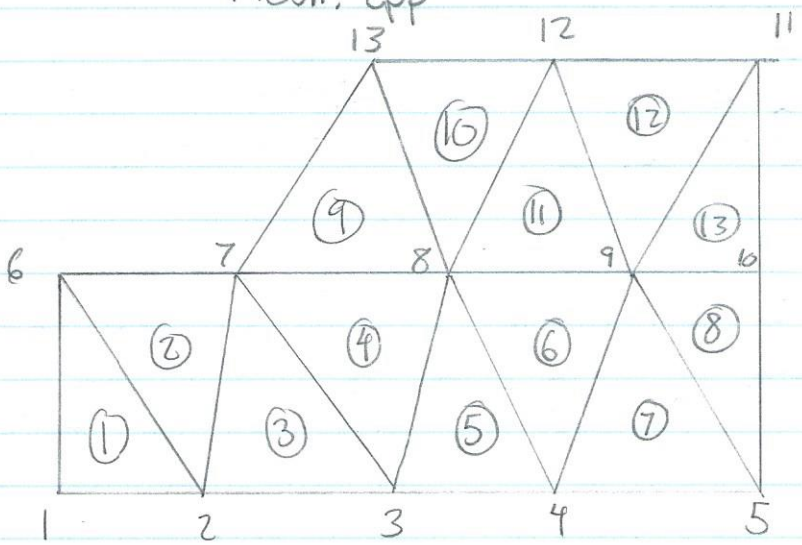
---

psup2 = [0 2 ~~3 6~~ 6

PSUP2 = [0 2 6 10 14 ...

Mesh.cpp

Ex.



CONNEC [NELEM][NNODE]

Indice			
0	1	2	6
1	2	7	6
2	2	3	7
3	3	8	7
4	3	4	8
5	4	9	8
6	4	5	9
7	5	10	9
8	7	8	13
9	8	12	13
10	8	9	12
11	9	11	12
12	9	10	11

13 éléments

3 points par élément.

NELEM = 13
NNODE = 3 *
NPOIN = 13

* NNODE est constant dans ce maillage.
↳ Il faudrait faire un vecteur qui stocke le nombre de noeuds par élément.

33;

mesup = 3·13 = NNODE * NELEM = 39

* sum of NNODE si nrelem.

1. Elements surrounding points – Linked lists

esup1 [mesup] → stores the elements } linked lists

esup2 [NPOIN+1] → stores the location } linked lists

esup1 = [1 | 1 2 3 | 3 4 5 | 5 6 7 | 7 8 | 1 2 | 2 3 4 9 |
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17

4 5 6 9 10 11 | 6 7 8 11 12 13 | 8 13 | 12 13 | 10 11 12 | 9 10]
18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38

esup2 = [0 1 4 7 10 12 14 18 24 30 32 34 37 39]
0 1 2 3 4 5 6 7 8 9 10 11 12 13

mpsup est = à 50 ici, préférable d'utiliser un vecteur avec push_back.

2. Points surrounding points

psup1 [mpsup] → stores the points
psup2 [NPOIN + 1] → stores the locations

lpoin [NPOIN] → helps avoid repetition

Algorithm:

```
lpoin[NPOIN] = {0}
psup2(1) = 0
istor = 0

do ipoin=1, npoin
   do iesup = esup2(ipoin)+1, esup2(ipoin+1)
      ielem = esup1(iesup)
      do inode=1, nnode
         jpoin = inpoel(inode, ielem)
         if (jpoin ≠ ipoin and lpoin(jpoin) ≠ ipoin)
            istor = istor+1;
            psup1(istor) = jpoin
            lpoin(jpoin) = ipoin
         endif
      enddo
   enddo
   psup2(ipoin+1) = istor
enddo
```

Application de PSUP : Fortran

ipoin = 1 ⟶ (de 1 à 13)
  iesup = 1 ⟶ (de 1 à 1 par ipoin = 1)
  iclem = 1
    inode = 1
      jpoin = 1
      if (jpoin ≠ 1 & 0 ≠ 1) NON

    inode = 2
      jpoin = 2
      if (2 ≠ 1 & 0 ≠ 1) oui
        istor = 1
        psup1(1) = 2 → psup1 = [2 0 0...
        lpoin(2) = 1 lpoin = [0 1 0 0...

    inode = 3
      jpoin = 6
      if (6 ≠ 1 & 0 ≠ 1) oui
        istor = 2
        psup1(2) = 6 ⟶ psup1 = [2 6 0...]
        lpoin(6) = 1 → lpoin = [0 1 0 0 0 1 0...]

psup2(1+1) = 2 ⟶ psup2 = [0 2 0...]

C++

ipoin = 0
  iesup = 0
    ielem = 0
    inode = 0
      jpoin = 1
      if (1 ≠ 1 & 0 ≠ 1) NON

    inode = 1
      jpoin = 2
      if (2 ≠ 1 & 0 ≠ 1) oui
        istor = 1 ;
        psup1(0) = 2
        lpoin(1) = 1

    inode = 2
      jpoin = 6
      if (6 ≠ 1 & 0 ≠ 1) oui
        istor = 2
        psup1(1) = 6
        lpoin(5) = 1

psup2(1) = 2

⇒ psup1 = [2 6]
  lpoin = [0 1 0 0 0 1 0 ...]

psup2 = [0 2 0 ...]

NPSUP à partir de psup2 :

NPSUP [NPOIN]

3. Elements surrounding elements. (ESUEL)

ESUEL [NELEM][NFAEL]

↳ number of faces per element.

Dans l'exemple, NFAEL = 3

* Faudrait créer un vecteur qui store cette variable par chaque élément.

```
lpoin [NPOIN] = {0}
ESUEL [NELEM][NFAEL] = {0}

do  ielem = 1, nelem
    do ifael = 1, nfael
       nnofa = lnofa (ifael)
       lhelp (1:nnofa) = inpoel (lpofa (1:nnofa, ifael), ielem)
       lpoin (lhelp (1:nnofa)) = 1
       ipoin = lhelp (1)
       do  istor = esup2 (ipoin) + 1, esup2 (ipoin + 1)
           jelem = esup1 (istor)
           if (jelem. ≠ ielem)
              do  jfael = 1, nfael
                  nnofj = lnofa (jfael)
                  if (nnofj = nnofa)
                     icun = 0
                     do  jnofa = 1, nnofa
                         jpoin = inpoel (lpofa (jnofa, jfael), jelem
                         icun = icun + lpoin (jpoin)
                  enddo
                  if (icun = nnofa)
                     esuel (ifael, ielem) = jelem
                  endif
              endif
           enddo
       endif
    enddo
lpoin (lhelp (1:nnofa)) = 0
```

* Je crois qu'il manque un enddo ici

3 enddo

fortran { lhelp(1:nnofa), lnofa(1:nfael), lpofa(1:nnofa, 1:nfael)

Initialize:

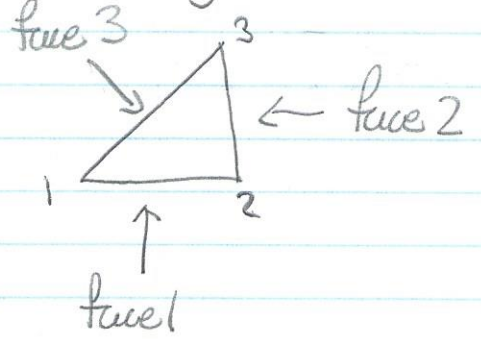
lpain[NPOIN] = {0}
ESUEL[NELIEM][NFAEL] = {0}

* Par un code 2D, nnofa = 2 (lignes)
↳ number of nodes per face.

nnofa => number of nodes per face = 2

nfael => number of faces per element
↳ par des éléments triangulaires (2D) = 3

Ex: 2D (triangle)



* lpofa est une matrice prédéfinie qui dépend de la forme des éléments.
↳ On peut tout de suite créer une telle matrice par différents types d'éléments

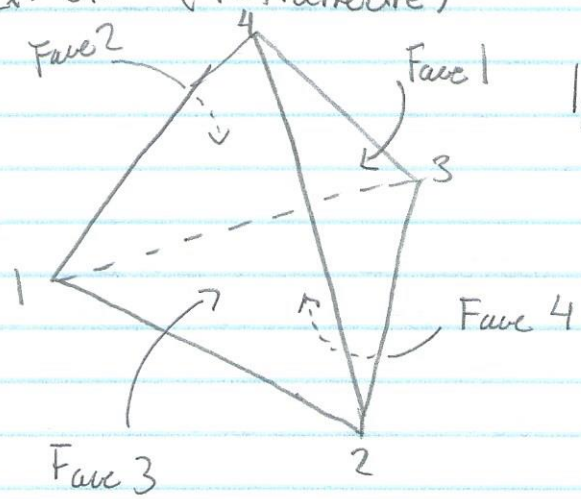
lpofa => list (Matrix) of points per face

lpofa[nfael][nnofa] ↳ lpofa[3][2] (dimensions) = $\begin{bmatrix} 1 & 2 \\ 2 & 3 \\ 3 & 1 \end{bmatrix}$ → face 1, face 2, face 3

lnofa => list of nodes per face lnofa[nfael]

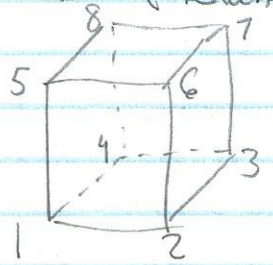
↳ lnofa[3] = [2 2 2]
(3 : dim)

Ex: 3D (tétrahèdre)



$$\text{Ipofa}[4][3] = \begin{bmatrix} 2 & 3 & 4 \\ 3 & 1 & 4 \\ 1 & 2 & 4 \\ 1 & 3 & 2 \end{bmatrix} \begin{matrix} \rightarrow \text{Face 1} \\ \rightarrow \text{Face 2} \\ \rightarrow \text{Face 3} \\ \rightarrow \text{Face 4} \end{matrix}$$

Ex: 3D (hexahèdre)



$$\text{Ipofa}[6][4] = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 1 & 2 & 6 & 5 \\ 2 & 3 & 7 & 6 \\ 3 & 4 & 8 & 7 \\ 4 & 1 & 5 & 8 \\ 5 & 6 & 7 & 8 \end{bmatrix}$$

CONNEC(1, [1 2]) = [1 2]

0
0→2
2

ielem = 1
  ifael = 1 (1→3) [1 2]
    nnofa = 2
    lhelp[1:2] = CONNEC(1, lpofa(1, 1:2)) => lhelp = [1 2]
    lpoin([1 2]) = 1
    ipoin = 1
    istor = 1 (début du do) 1→1
      jelem = 1
      if(1 ≠ 1) NON
        ↳ ne rentre pas dans le if
      endif
    enddo
    lpoin([1 2]) = 0 => Reset lpoin

  ifael = 2 [2 3]
    nnofa = 2
    lhelp[1 2] = CONNEC(1, lpofa(2, [1 2])) => lhelp = [2 6]
    lpoin[2 6] = 1 => lpoin = [0 1 0 0 0 1]
    ipoin = 2
    istor = 2 (2→4)
      jelem = 1
      if(1 ≠ 1) NON → ne rentre pas dans le if
      endif

lhelp=[2 6]
lpoin[1 5]=1
ipoin = 2
↳3

    istor = 3
      jelem = 2 oui
      if(2 ≠ 1)
        jfael = 1 (for de 1→3)
          nnofj = lnofa(jfael) = 2
          if(2 == 2) oui
            icon = 0
            jnofa = 1 (for de 1→2)

(suite →)

ne pas oublier de ↷ ipola.

jnofa = 1

jpoin = CONNEC (2, ipola (1, 1)) = 2
icount = ipoint (jpoin) = 0+1 = 1

jnofa = 2

jpoin = CONNEC (2, ipola (1, 2)) = 7 } 6
icount = 1 + ipoin (7) = 1+0
6

if (1 == 2) non

esuel [1] [2] = 2

NON !

* Je crois que la table CONNEC doit se présenter avec les noeuds en ordre croissant.

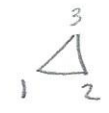
C'est plus qu'il doit y avoir un respect de l'ordre d'attribution des faces dans ipola ?

et fpola

ielem = 0 => élément 1
ifael = 0 => face 1

lnofa = [2 2 2]

lpofa = $\begin{bmatrix} 1 & 2 \\ 2 & 3 \\ 3 & 1 \end{bmatrix}$



face 1

nnofa = 2
lhelp = [1 2]
lpoin = [1 1 0 0 0 0 0 0 0 0 0 0 0 0]
ipoin = 1
istor = 0 For istor = 0 → 0 => ne rentre pas dans le for
⋮
lpoin = [0 0 0 0 0 0 0 0 0 0 0 0 0 0] remise à 0.

ifael = 1 => face 2
nnofa = 2
lhelp = [2 6]
lpoin = [0 1 0 0 0 1 0 0 0 0 0 0 0 0]
ipoin = 2
istor = 1 (For istor = 1 → 3)
jelem = 1
if (1-1 ≠ 0) faux
⋮

On regarde l'élément 1 par rapport à l'élément 1', on s'en fout!

istor = 2
jelem = 2
if (2-1 ≠ 0) vrai → 1 ≠ 0
jfael = 0 (For jfael = 0 → 2)
nnofj = 2 nombre de noeuds de la face "j"
if (2 == 2) vrai
icun = 0
jnofa = 0 (For jnofa = 0 → 1)
jpoin = 2
icun = icun + lpoin [jpoin] = 0 + 1 = 1

jnofa = 1
jpoin = 6 * si CONNEC croissant *
icun = 1 + 1 = 2
if (2 == 2)
esuel [0][1] = 2

Il faut regarder TOUTES LES COMBINAISONS POSSIBLES!!