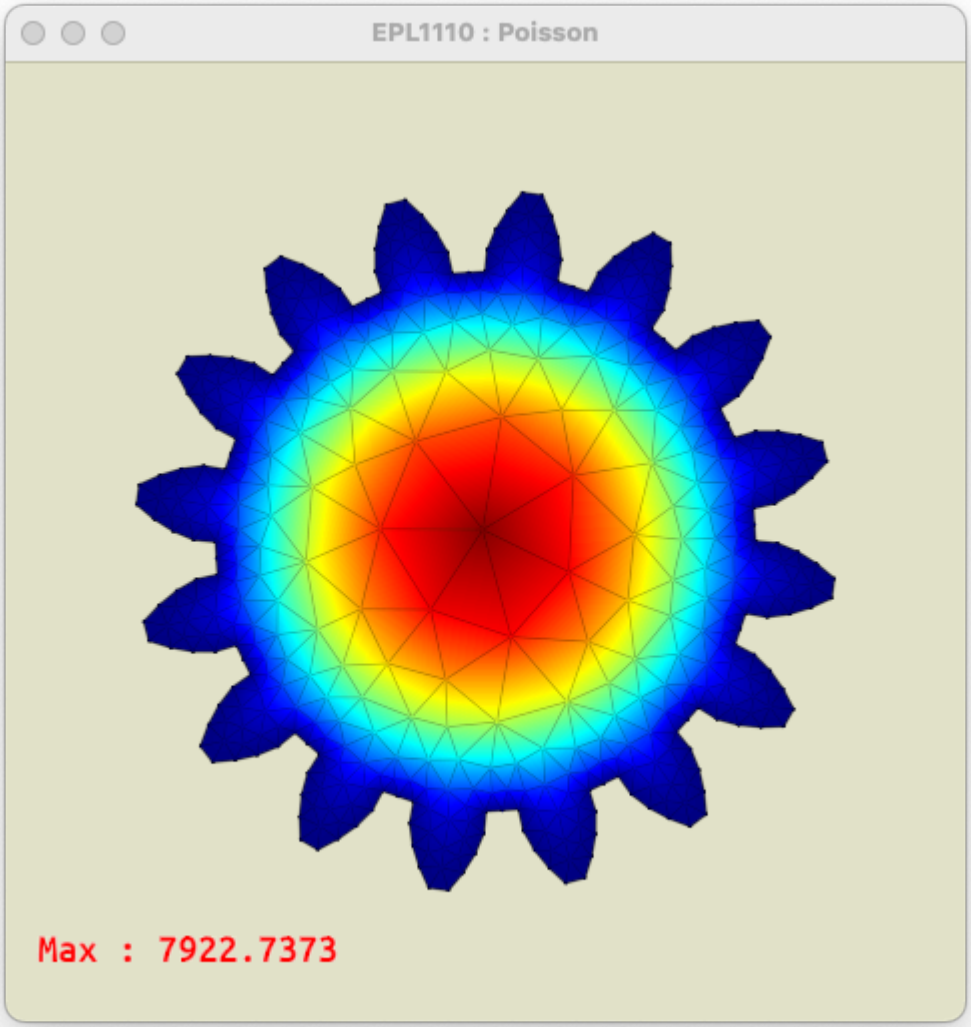
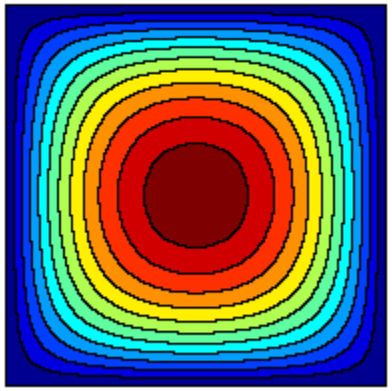


Nous allons faire notre premier vrai programme d'éléments finis !

Un exemple tout simple et bien connu :-)

$$\nabla^2 u(x,y) + 1 = 0, \quad (x,y) \in \Omega,$$

$$u(x,y) = 0, \quad (x,y) \in \partial\Omega,$$



$$u(x,y) = \sum_{i,j \text{ odd}} C_{ij} \sin\left(\frac{i\pi(x+1)}{2}\right) \sin\left(\frac{j\pi(y+1)}{2}\right)$$

Solution analytique

$$\nabla^2 u(x,y) = -1,$$

$$\sum_{i,j \text{ odd}} \pi^2(i^2+j^2) C_{ij} \sin\left(\frac{i\pi(x+1)}{2}\right) \sin\left(\frac{j\pi(y+1)}{2}\right) = 1,$$

Because sin are orthogonal,

$$\frac{\pi^2(i^2+j^2)}{4} C_{ij} = \underbrace{\int_\Omega \sin\left(\frac{i\pi(x+1)}{2}\right) \sin\left(\frac{j\pi(y+1)}{2}\right) d\Omega}_{16/(ij\pi^2)},$$

$$u(x,y) = \sum_{i,j \text{ odd}} C_{ij} \sin\left(\frac{i\pi(x+1)}{2}\right) \sin\left(\frac{j\pi(y+1)}{2}\right)$$

On a ici une solution analytique !
Mais, ce n'est pas le cas dans la plupart des vrais problèmes.
Il s'agit juste d'un exemple pour présenter notre méthode !

Construction du système linéaire discret

$$J(u^h) = \frac{1}{2}\int_\Omega (\nabla u^h)\cdot(\nabla u^h)\, d\Omega - \int_\Omega f\, u^h\, d\Omega,$$

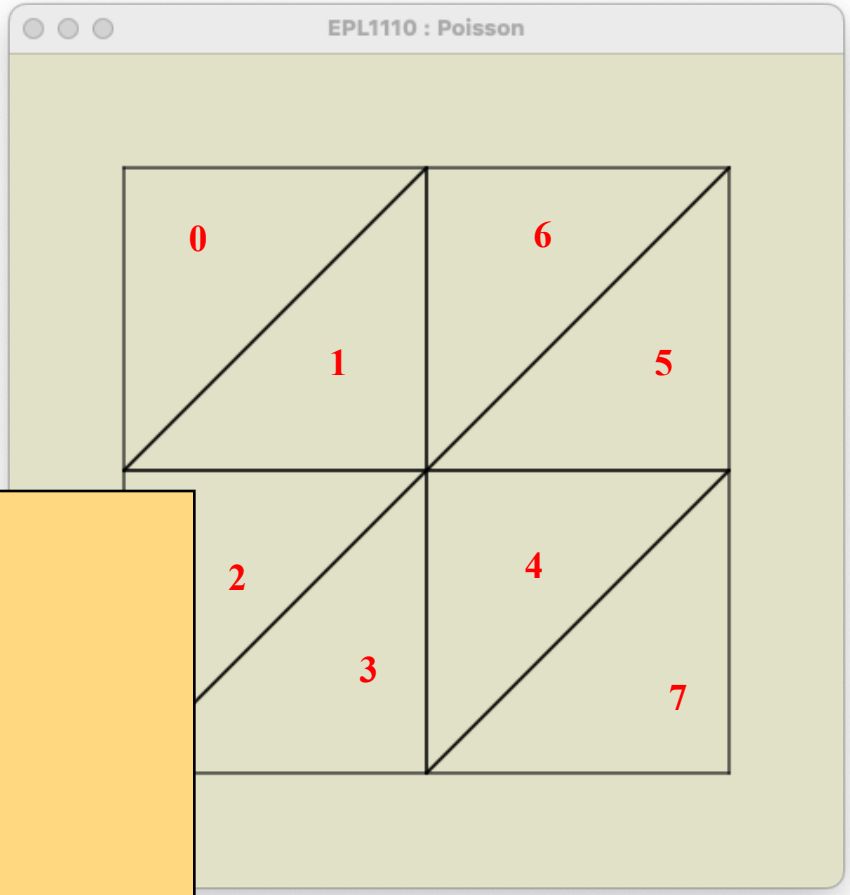
$$J(u^h) = \frac{1}{2}\int_\Omega \left(\sum_{i=1}^{n} U_i \nabla\tau_i\right)\cdot\left(\sum_{j=1}^{n} U_j \nabla\tau_j\right) d\Omega - \int_\Omega f\, \left(\sum_{i=1}^{n} U_i\tau_i\right) d\Omega,$$

$$J(u^h) = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} U_i\, U_j \boxed{\int_\Omega (\nabla\tau_i)\cdot(\nabla\tau_j)\, d\Omega} - \sum_{i=1}^{n} U_i \boxed{\int_\Omega f\, \tau_i\, d\Omega},$$

$$J(u^h) = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} U_i\, U_j \boxed{A_{ij}} - \sum_{i=1}^{n} U_i \boxed{B_i},$$

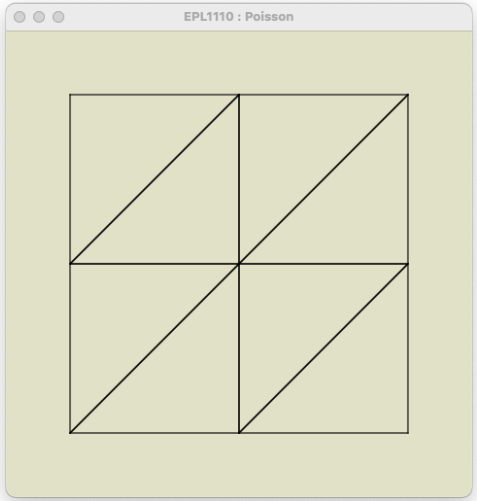
$$0 = \frac{\partial J(u^h)}{\partial U_i} = \sum_{j=1}^{n} \boxed{A_{ij}}\, U_j - \boxed{B_i}, \quad i = 1, n.$$

Faisons un petit exemple !

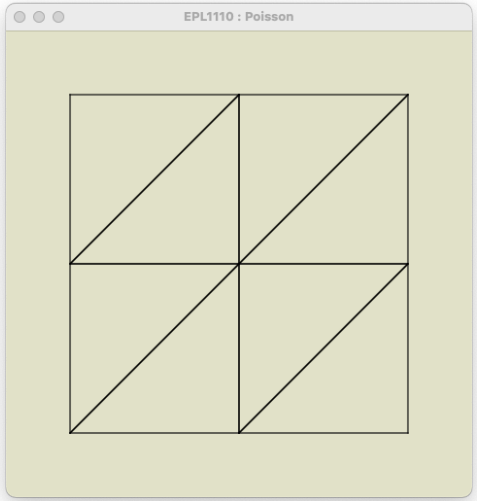



```
Number of nodes 9
     0 :  0.0000000e+00  2.0000000e+00
     1 :  1.0000000e+00  2.0000000e+00
     2 :  2.0000000e+00  2.0000000e+00
     3 :  0.0000000e+00  1.0000000e+00
     4 :  1.0000000e+00  1.0000000e+00
     5 :  2.0000000e+00  1.0000000e+00
     6 :  0.0000000e+00  0.0000000e+00
     7 :  1.0000000e+00  0.0000000e+00
     8 :  2.0000000e+00  0.0000000e+00
Number of triangles 8
     0 :       3       1       0
     1 :       3       4       1
     2 :       6       4       3
     3 :       6       7       4
     4 :       7       5       4
     5 :       4       5       2
     6 :       4       2       1
     7 :       7       8       5
```

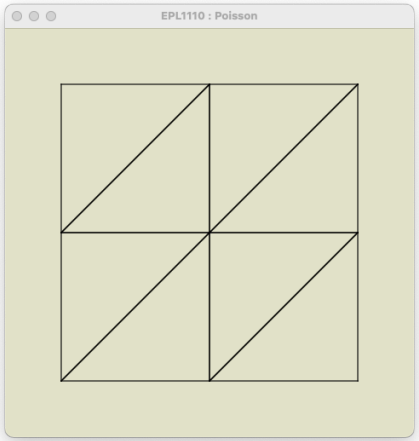
Deux matrices locales



Deux matrices locales



Assemblons les matrices

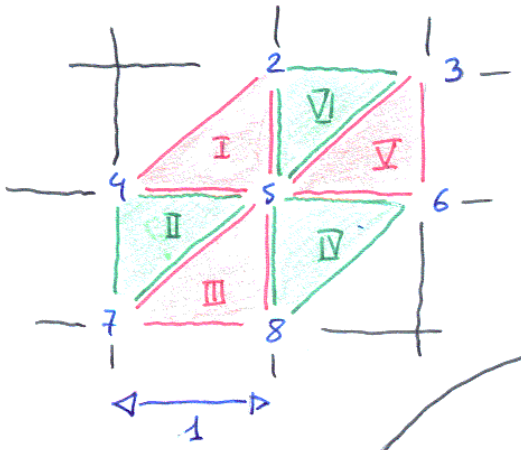



```
0 :  3  1  0
1 :  3  4  1
2 :  6  4  3
3 :  6  7  4
4 :  7  5  4
5 :  4  5  2
6 :  4  2  1
7 :  7  8  5
```

STIFFNESS MATRIX CALCULATION

$$\left\langle \frac{\partial \phi_i^e}{\partial x}\frac{\partial \phi_j^e}{\partial x} + \frac{\partial \phi_i^e}{\partial y}\frac{\partial \phi_j^e}{\partial y} \right\rangle_{\Omega_e}$$

$$A_{ij} = \bigoplus_{e} A_{ij}^{e}$$



CHECK !

$\sum \phi_i = 1$

$\sum \phi_{i,x} = 0$

$\sum \phi_{i,y} = 0$

$\phi_1 = (1-x) + \alpha t$

$\phi_{1,x} = -1$

$\phi_{1,y} = 0$

	$\phi_{i,x}$	$\phi_{i,y}$
1	-1	0
2	1	-1
3	0	1

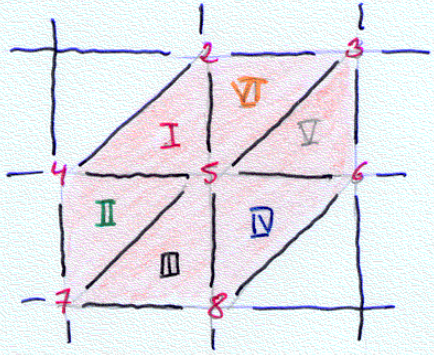
$$\frac{1}{2}\begin{bmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{bmatrix}$$

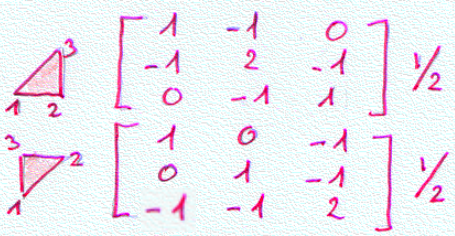
A_{ij}^{e}

	$\phi_{i,x}$	$\phi_{i,y}$
1	0	-1
2	1	0
3	-1	1

$$\frac{1}{2}\begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & -1 \\ -1 & -1 & 2 \end{bmatrix}$$

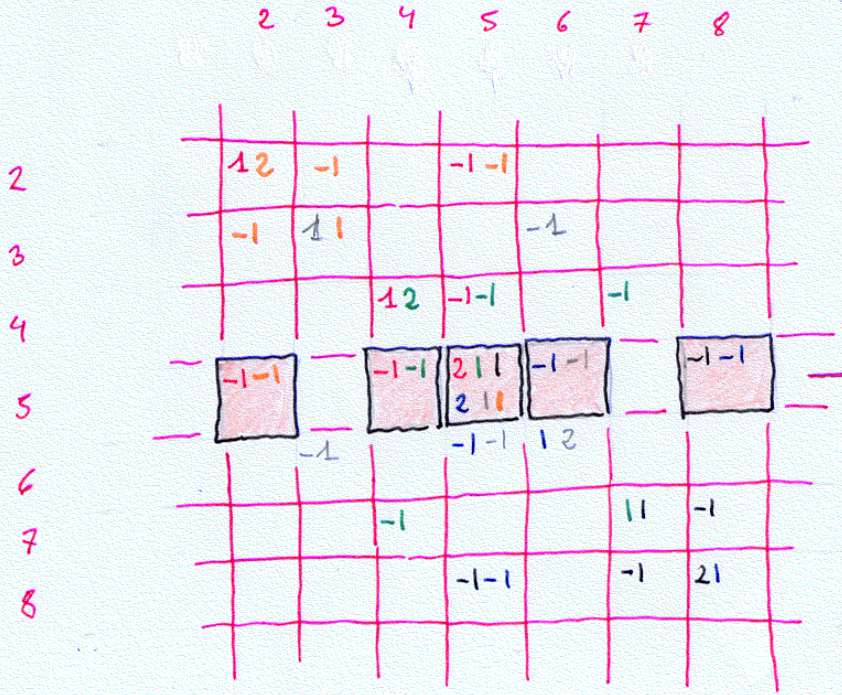
ASSEMBLING LOCAL MATRICES



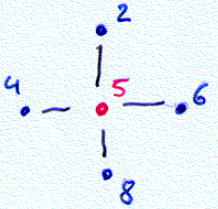


LOCATION TABLE

I	4	5	2
II	7	5	4
III	7	8	5
IV	8	6	5
V	5	6	3
VI	5	3	2



$$\frac{1}{2}\left(8U_5 - 2U_2 - 2U_4 - 2U_8 - 2U_6\right)$$



REGULAR LATTICE OF TURNER TRIANGLES = CENTRAL FINITE DIFFERENCES

Une structure pour un système linéaire Ax = b

```
typedef struct {
    double **A;
    double  *B;
    int size;
} femFullSystem;
```

```
theSystem = femFullSystemCreate(2);
double **A = theSystem->A;
double  *B = theSystem->B;
A[0][0] = 1.0; A[0][1] = 1.0; B[0] = 4;
A[1][0] = 1.0; A[1][1] = 2.0; B[1] = 7;
femFullSystemEliminate(theSystem);
femFullSystemPrint(theSystem);
```

Nous allons faire les tâches suivantes :

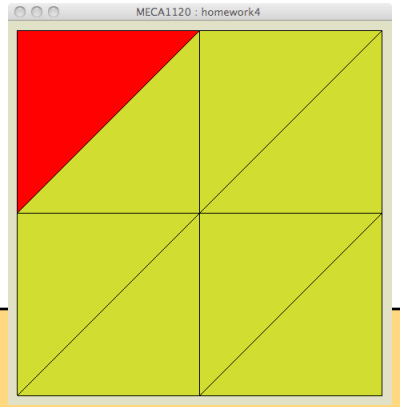
- **Allouer la matrice et le vecteur**
- **Initialiser le système**
- **Imprimer le système**
- **Contraindre une valeur**
- **Résoudre le système**

```
+1.0e+00 +1.0e+00 :   +1.0e+00
+1.0e+00 +1.0e+00 :   +3.0e+00
```

Résolution « en place » par élimination de Gauss de matrices symétriques définies positives

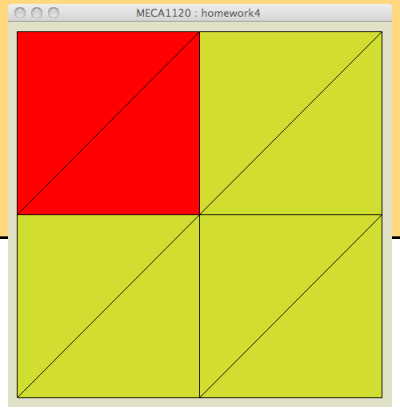
```
femFullSystem* femFullSystemCreate(int size);
void femFullSystemPrint(femFullSystem* mySystem);
void femFullSystemConstrain(femFullSystem* mySystem, int myNode, double myValue);
void femFullSystemEliminate(femFullSystem* mySystem);
```

Assemblage élément…



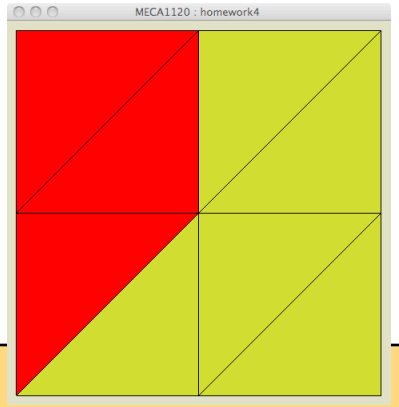

```
+1.0e+00 -5.0e-01          -5.0e-01                                  :  +1.7e-01
-5.0e-01 +5.0e-01                                                    :  +1.7e-01
                                                                     :  +0.0e+00
-5.0e-01                   +5.0e-01                                  :  +1.7e-01
                                                                     :  +0.0e+00
                                                                     :  +0.0e+00
                                                                     :  +0.0e+00
                                                                     :  +0.0e+00
                                                                     :  +0.0e+00
```

```
+1.0e+00 -5.0e-01          -5.0e-01                                  :  +1.7e-01
-5.0e-01 +1.0e+00                    -5.0e-01                        :  +3.3e-01
                                                                     :  +0.0e+00
-5.0e-01                   +1.0e+00 -5.0e-01                         :  +3.3e-01
         -5.0e-01          -5.0e-01 +1.0e+00                         :  +1.7e-01
                                                                     :  +0.0e+00
                                                                     :  +0.0e+00
                                                                     :  +0.0e+00
                                                                     :  +0.0e+00
```

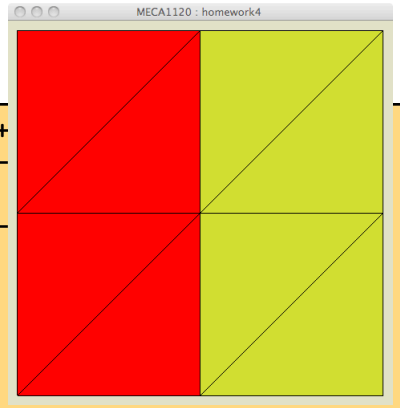


…par élément

Assemblage élément…



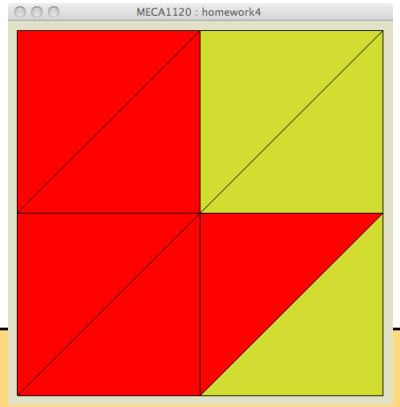

```
+1.0e+00 -5.0e-01          -5.0e-01                                  :  +1.7e-01
-5.0e-01 +1.0e+00                    -5.0e-01                        :  +3.3e-01
                                                                     :  +0.0e+00
-5.0e-01                   +2.0e+00 -1.0e+00          -5.0e-01       :  +5.0e-01
         -5.0e-01          -1.0e+00 +1.5e+00                         :  +3.3e-01
                                                                     :  +0.0e+00
                           -5.0e-01                   +5.0e-01       :  +1.7e-01
                                                                     :  +0.0e+00
                                                                     :  +0.0e+00
```



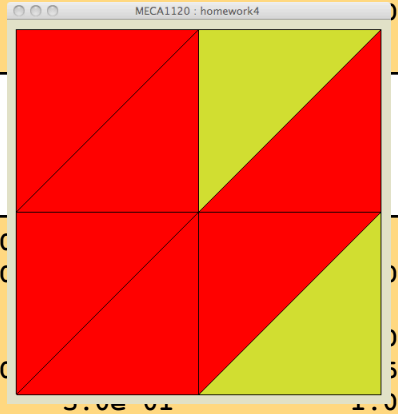

```
+                          -5.0e-01                                  :  +1.7e-01
-                                    -5.0e-01                        :  +3.3e-01
                                                                     :  +0.0e+00
-                          +2.0e+00 -1.0e+00          -5.0e-01       :  +5.0e-01
                           -1.0e+00 +2.0e+00                -5.0e-01 :  +5.0e-01
                                                                     :  +0.0e+00
                           -5.0e-01                   +1.0e+00 -5.0e-01 :  +3.3e-01
                                    -5.0e-01          -5.0e-01 +1.0e+00 :  +1.7e-01
                                                                     :  +0.0e+00
```

…par élément

Assemblage élément…



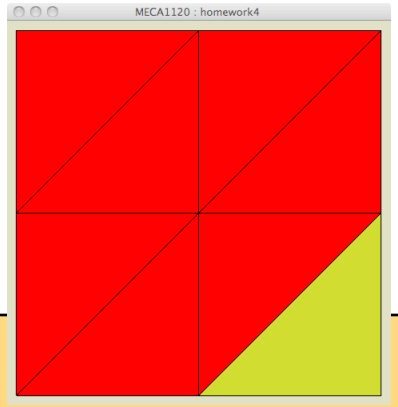

```
+1.0e+00 -5.0e-01          -5.0e-01                                          +1.7e-01
-5.0e-01 +1.0e+00                   -5.0e-01                                 +3.3e-01
                                                                          :  +0.0e+00
-5.0e-01                   +2.0e+00 -1.0e+00          -5.0e-01              :  +5.0e-01
         -5.0e-01          -1.0e+00 +3.0e+00 -5.0e-01          -1.0e+00     :  +6.7e-01
                                    -5.0e-01 +5.0e-01                     :  +1.7e-01
                           -5.0e-01                   +1.0e+00 -5.0e-01     :  +3.3e-01
                                                      -5.0e-01 +1.5e+00     :  +3.3e-01
                                                                          :  +0.0e+00
```




```
+1.0e+00 -5.0e-                                                           :  +1.7e-01
-5.0e-01 +1.0e+0                    )e-01                                 :  +3.3e-01
                                          -5.0e-01                        :  +1.7e-01
-5.0e-01                            )e+00          -5.0e-01               :  +5.0e-01
         -5.0e-                     5e+00 -1.0e+00          -1.0e+00      :  +8.3e-01
                                   1.0e+00 +1.5e+00                       :  +3.3e-01
                           -5.0e-01                   +1.0e+00 -5.0e-01     :  +3.3e-01
                                    -1.0e+00          -5.0e-01 +1.5e+00     :  +3.3e-01
                                                                          :  +0.0e+00
```

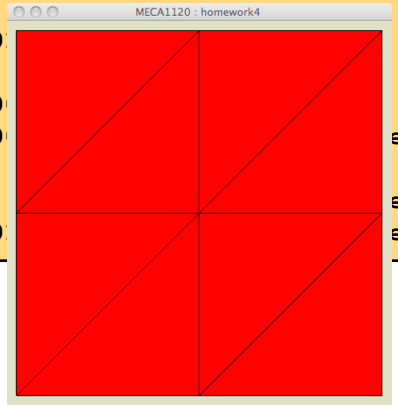
…par élément

Assemblage élément…




```
+1.0e+00 -5.0e-01          -5.0e-01                                        :  +1.7e-01
-5.0e-01 +2.0e+00 -5.0e-01          -1.0e+00                               :  +5.0e-01
         -5.0e-01 +1.0e+00                    -5.0e-01                    :  +3.3e-01
-5.0e-01                   +2.0e+00 -1.0e+00          -5.0e-01            :  +5.0e-01
         -1.0e+00          -1.0e+00 +4.0e+00 -1.0e+00          -1.0e+00   :  +1.0e+00
                  -5.0e-01          -1.0e+00 +1.5e+00                     :  +3.3e-01
                            -5.0e-01                    +1.0e+00 -5.0e-01 :  +3.3e-01
                                      -1.0e+00          -5.0e-01 +1.5e+00 :  +3.3e-01
                                                                          :  +0.0e+00
```

```
+1.0e+00 -5.0e-01          -5.0e-01                                        :  +1.7e-01
-5.0e-01 +2.0e+00 -5.0e-01          -1.0e+00                               :  +5.0e-01
         -5.0e-01 +1.0e+00                    -5.0e-0                     :  +3.3e-01
-5.0e-01                   +2.0e+00 -1.0e+00                              :  +5.0e-01
         -1.0e+00          -1.0e+00 +4.0e+00 -1.0e+0                      :  +1.0e+00
                  -5.0e-01          -1.0e+00 +2.0e+0                 e-01 :  +5.0e-01
                            -5.0e-01                                      :  +3.3e-01
                                      -1.0e+00                        e-01 :  +5.0e-01
                                                -5.0e-0               e+00 :  +1.7e-01
```



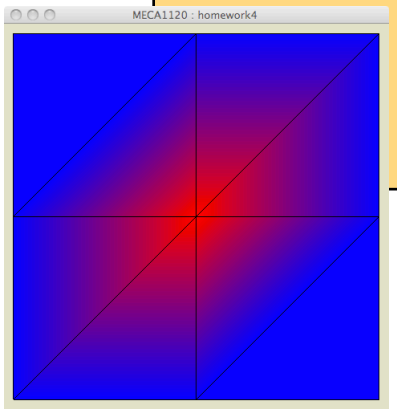
…par élément

Comment imposer les conditions frontières ?

```
def constrain(self,myNode,myValue):
    A = self.A; B = self.B; n = self.n
    for i in range(n):
      B[i] -= myValue * A[i,myNode]
      A[i,myNode] = 0
    for i in range(n):
      A[myNode,i] = 0
    A[myNode,myNode] = 1;
    B[myNode] = myValue;
```

```
for myEdge in theEdges.edges:
    if (myEdge[3] == -1) :
      theSystem.constrain(myEdge[0],0.0)
      theSystem.constrain(myEdge[1],0.0)
```

```
+1.0e+00                                                    :  +0.0e+00
        +1.0e+00                                            :  +0.0e+00
                +1.0e+00                                    :  +0.0e+00
                        +1.0e+00                            :  +0.0e+00
                                +4.0e+00                    :  +1.0e+00
                                        +1.0e+00            :  +0.0e+00
                                                +1.0e+00    :  +0.0e+00
                                                        +1.0e+00 :  +0.0e+00
                                                                +1.0e+00 :  +0.0e+00
```

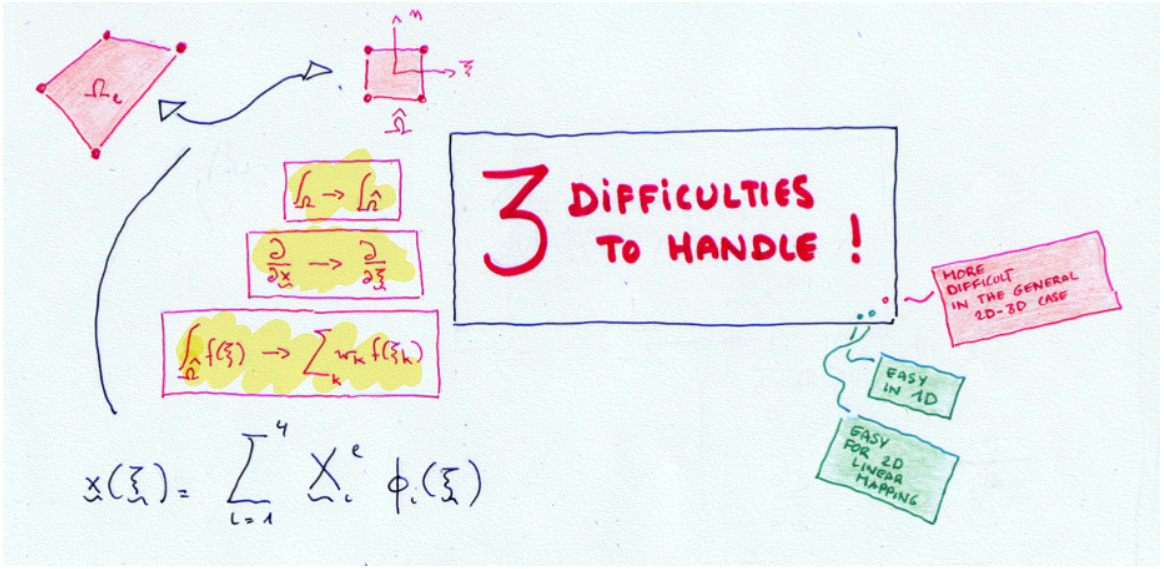


Les éléments triangulaires, c'est bien :-)
Mais, ici des éléments quadrilatères, ce serait mieux !

Et on fait comment pour des quadrilatères ?

$$A_{ij} = \int_\Omega (\nabla \tau_i) \cdot (\nabla \tau_j)\, d\Omega,$$

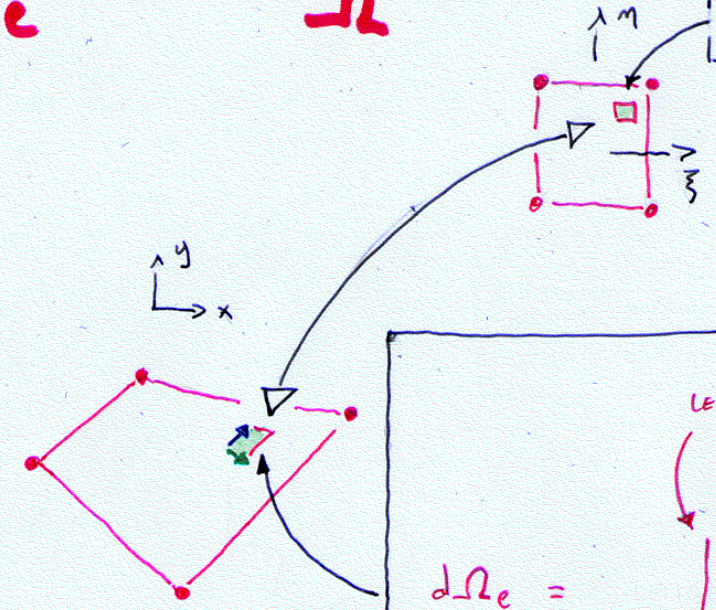
$$B_i = \int_\Omega f \tau_i \, d\Omega.$$



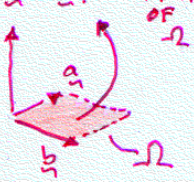
$$\int_{\Omega_\varepsilon} \to \int_{\widehat{\Omega}}$$

$\int_{\Omega_e} \rightarrow \int_{\hat{\Omega}}$

$d\hat{\Omega} = d\xi \, d\eta$



$|\underline{a} \times \underline{b}|$ = AREA OF Ω



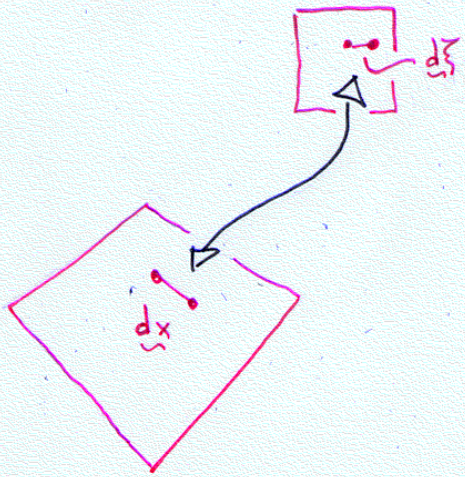
VECTORIAL PRODUCT

LENGTH

$$d\Omega_e = \left| \left(\frac{\partial \underline{x}}{\partial \xi} d\xi \times \frac{\partial \underline{x}}{\partial \eta} d\eta \right) \right|$$

$$= \underbrace{\left| \frac{\partial \underline{x}}{\partial \xi} \times \frac{\partial \underline{x}}{\partial \eta} \right|}_{\det\left(\frac{\partial \underline{x}}{\partial \xi \partial \eta}\right)} \underbrace{d\xi \, d\eta}_{d\hat{\Omega}}$$

Jc



$$\underline{x}(\underline{\xi}) = \sum_{i=1}^{4} \underline{X}_i^e \, \phi_i(\underline{\xi})$$

$$\underline{dx}(\underline{\xi}) = \left(\sum_{i=1}^{4} \underline{X}_i^e \, \frac{\partial \phi_i}{\partial \underline{\xi}}(\underline{\xi}) \right) \cdot \underline{d\xi}$$

$$\begin{bmatrix} dx \\ dy \end{bmatrix}$$

$$\begin{bmatrix} d\xi \\ d\eta \end{bmatrix}$$

$$\frac{\partial \underline{x}}{\partial \underline{\xi}} = \begin{bmatrix} \frac{\partial x}{\partial \xi} & \frac{\partial x}{\partial \eta} \\ \frac{\partial y}{\partial \xi} & \frac{\partial y}{\partial \eta} \end{bmatrix}$$

How to calculate J_e ?

$$J^e = \det\left(\frac{\partial \underline{x}}{\partial \underline{\xi}} \right)$$

$$\frac{\partial}{\partial x} \rightarrow \frac{\partial}{\partial z}$$

$\partial/\partial x \rightarrow \partial/\partial \xi$

IT IS NOT NECESSARY TO CALCULATE $\xi(x)$!

$\nabla \phi_i$

$$\frac{\partial \phi_i}{\partial x} = \frac{\partial \phi_i}{\partial \xi} \cdot \frac{\partial \xi}{\partial x}$$

$$\frac{\partial \phi_i}{\partial \xi} = \frac{\partial \phi_i}{\partial x} \cdot \frac{\partial x}{\partial \xi} = \sum_{i=1}^{4} x_i^e \frac{\partial \phi_i}{\partial \xi}$$

$$\left(\cdot\right)^{-1}$$

$$\begin{bmatrix} \partial\xi/\partial x & \partial\xi/\partial y \\ \partial\eta/\partial x & \partial\eta/\partial y \end{bmatrix} = \frac{1}{J_e} \begin{bmatrix} \partial y/\partial \eta & -\partial x/\partial \eta \\ -\partial y/\partial \xi & \partial x/\partial \xi \end{bmatrix}$$

$$A^{e}_{ij}$$

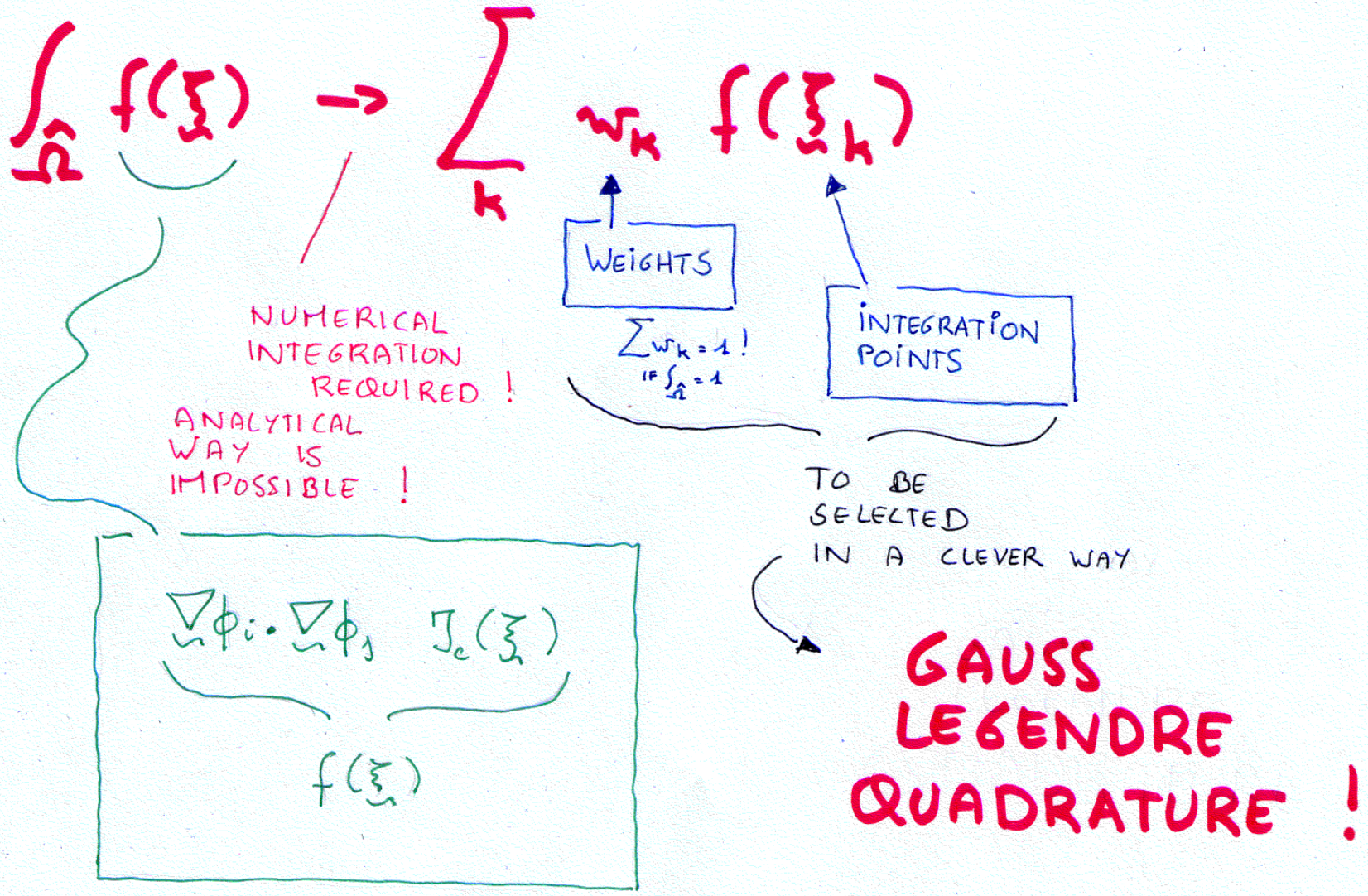
$$\begin{bmatrix} \frac{\partial \phi_i}{\partial x} & \frac{\partial \phi_i}{\partial y} \end{bmatrix} = \begin{bmatrix} \frac{\partial \phi_i}{\partial \xi} & \frac{\partial \phi_i}{\partial \eta} \end{bmatrix} \cdot \begin{bmatrix} \frac{\partial \xi}{\partial x} & \frac{\partial \eta}{\partial x} \\ \frac{\partial \xi}{\partial y} & \frac{\partial \eta}{\partial y} \end{bmatrix}$$

$$\sum Y_i \frac{\partial \phi_i}{\partial \eta} \qquad \frac{\partial \phi_i}{\partial x} = \frac{1}{J_e} \left(\frac{\partial y}{\partial \eta} \frac{\partial \phi_i}{\partial \xi} - \frac{\partial y}{\partial \xi} \frac{\partial \phi_i}{\partial \eta} \right) \qquad \sum Y_i \frac{\partial \phi_i}{\partial \xi}$$

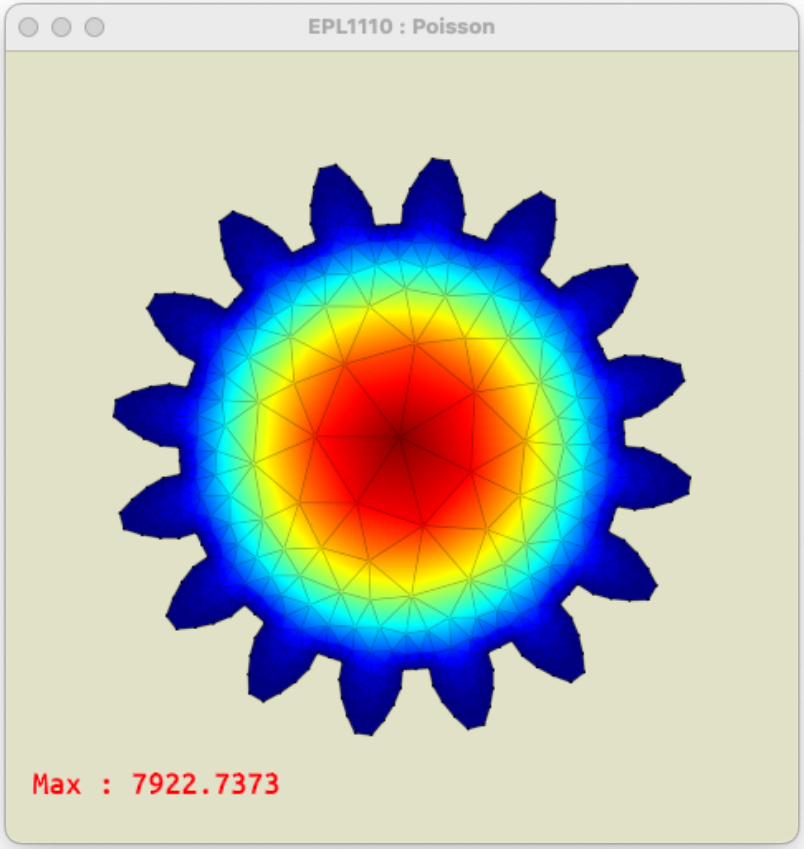
$$\text{FCT}(\xi, \eta)$$

$$A_{ij}^e = \int_{\hat{\Omega}} \underbrace{\underbrace{\nabla \phi_i \cdot \nabla \phi_j}_{\text{FCT}(\xi,\eta)} \; J_e(\xi, \eta)}_{\text{FCT}(\xi,\eta)} \, d\hat{\Omega}$$

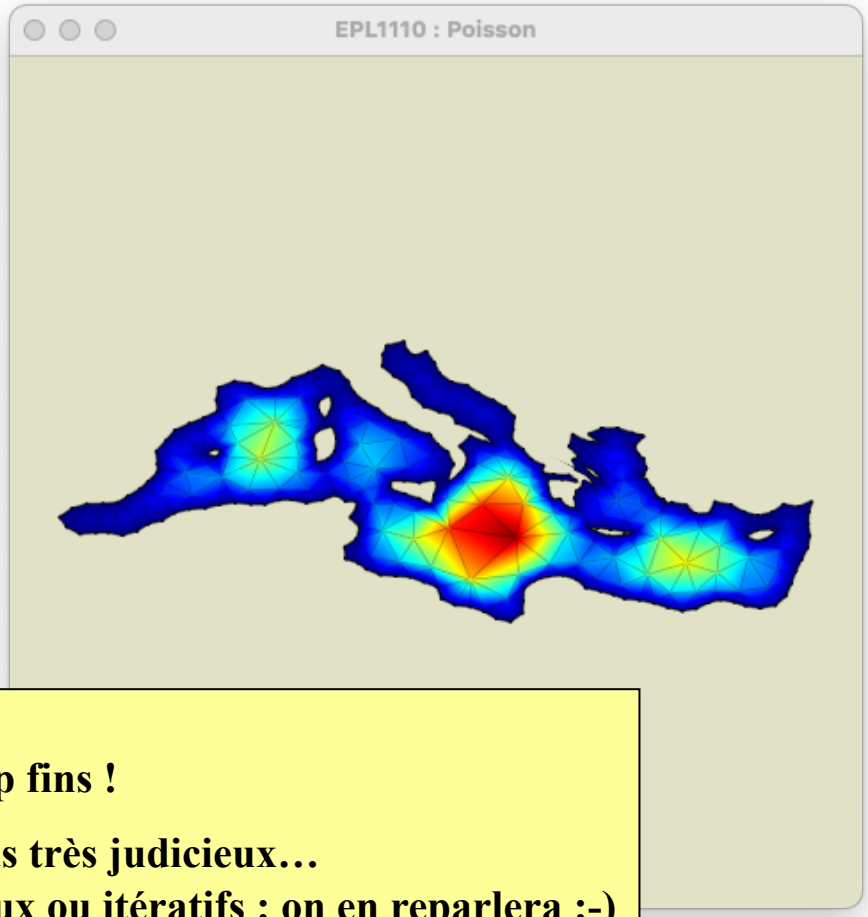
YOU HAVE JUST TO INTEGRATE NOW !


$\int_{\hat{\Omega}} f(\xi) \rightarrow \sum_k w_k f(\xi_k)$
NUMERICAL INTEGRATION REQUIRED !
ANALYTICAL WAY IS IMPOSSIBLE !
WEIGHTS
$\sum w_k = 1$! IF $\int_{\hat{\Omega}} = 1$
INTEGRATION POINTS
TO BE SELECTED IN A CLEVER WAY
$\nabla \phi_i \cdot \nabla \phi_j \; J_e(\xi)$
$f(\xi)$
GAUSS LEGENDRE QUADRATURE !

Et zou
On est prêt !



Calculer la solution du problème de Poisson avec des triangles linéaires ou des quads bilinéaires… Le jacobien n'est pas toujours constant !
Une code unique pour les deux cas !



Le code est très très lent…
Ne pas essayer des maillages trop fins !

Utiliser un solveur plein n'est pas très judicieux…
On fera appel à des solveurs creux ou itératifs : on en reparlera :-)

Un petit Poisson perdu dans la Méditerranée…