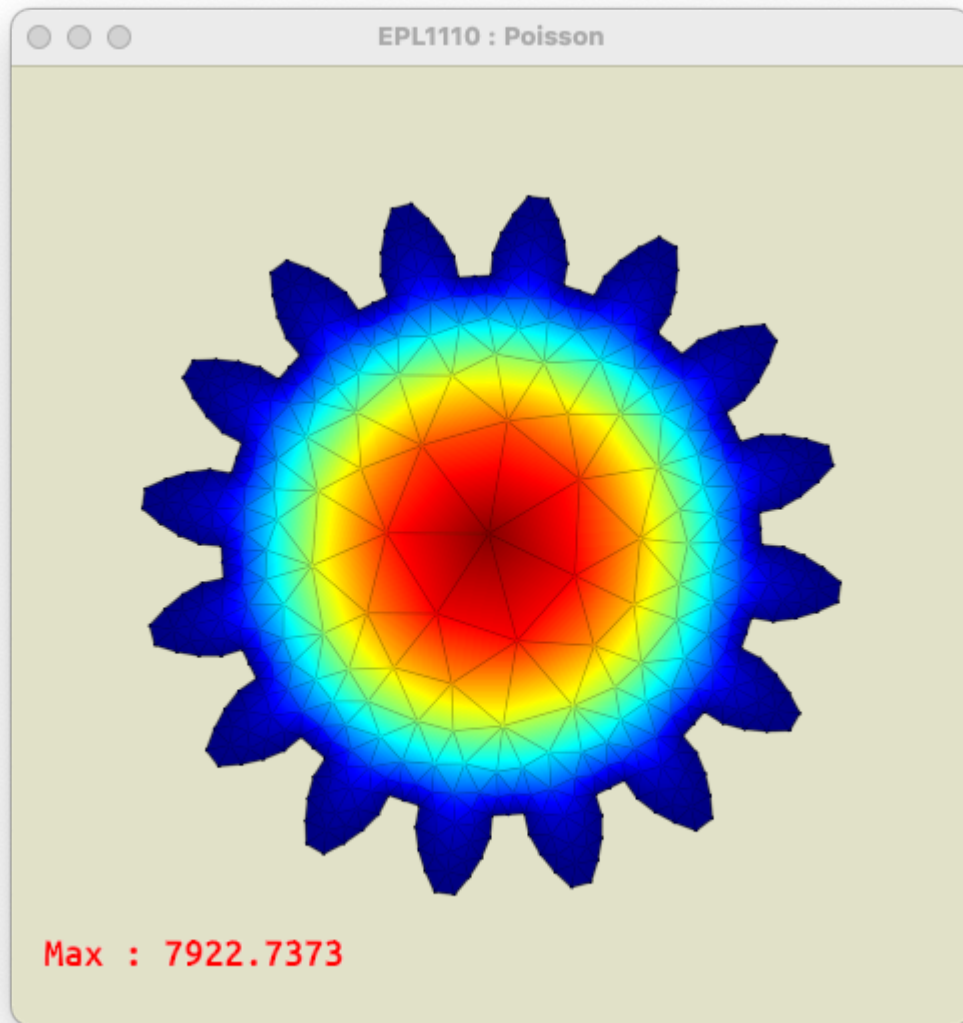


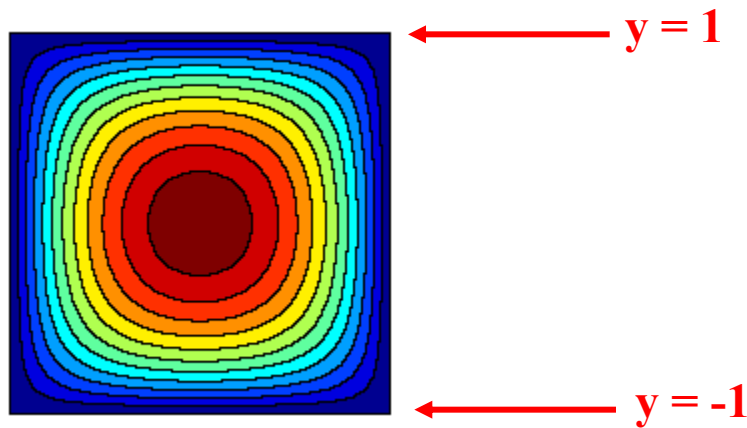


$$\begin{aligned} \nabla^2 u(x,y) + 1 &= 0 & \text{sur } \Omega \\ u(x,y) &= 0 & \text{sur } \partial\Omega \end{aligned}$$

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

Un
exemple
tout simple
et bien
connu :-)

$$\begin{aligned}\nabla^2 u(x, y) + 1 &= 0, & (x, y) &\in \Omega, \\ u(x, y) &= 0, & (x, y) &\in \partial\Omega,\end{aligned}$$



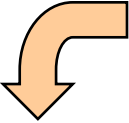
$$u(x, y) = \sum_{i,j \text{ odd}} C_{ij} \sin\left(\frac{i\pi(x+1)}{2}\right) \sin\left(\frac{j\pi(y+1)}{2}\right)$$

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

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

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

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

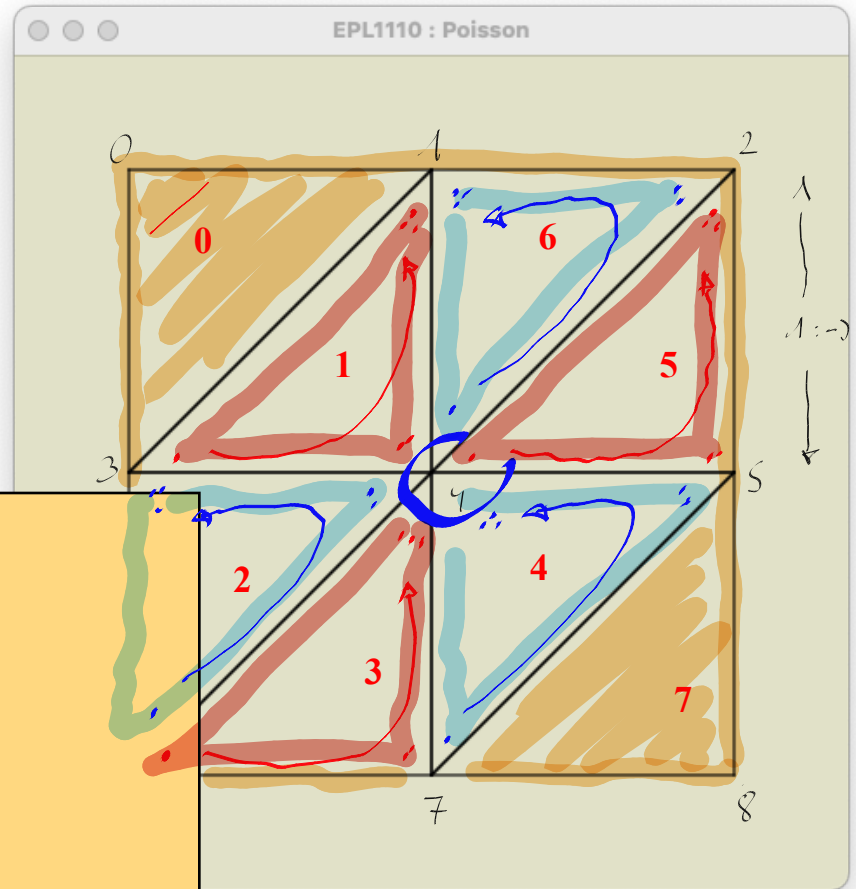
$$J(u^h) = \frac{1}{2} \sum_{i=1}^n \sum_{j=1}^n U_i U_j A_{ij} - \sum_{i=1}^n U_i B_i,$$

$$u \simeq u^h = \sum U_i \tau_i(x, y)$$



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

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

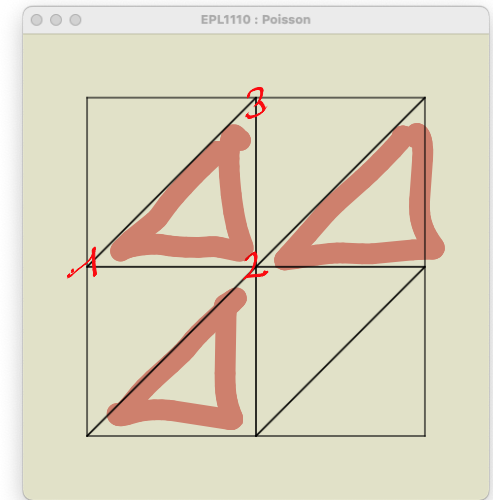
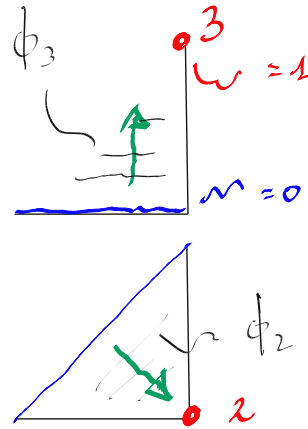
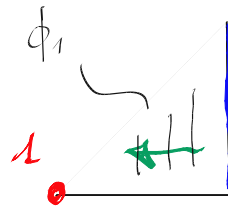
0 :	3	1	0
1 :	3	4	1
2 :	6	4	3
3 :	6	7	4
4 :	7	5	4
5 :	4	5	2
6 :	4	2	1
7 :	7	8	5

TABIEAU
D'APPARTEN.

$$\underbrace{A_{ij}}_{\text{MATRICE GLOBALE}} = \bigoplus \underbrace{A_{ij}^e}_{\text{MATRICES LOCALES}}$$

ASSEMBLAGE

Deux matrices locales



$$A_{ij}^e = \int_{\Omega_e} \phi_{i,x}^e \phi_{j,x}^e + \phi_{i,y}^e \phi_{j,y}^e \, dx dy$$

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

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

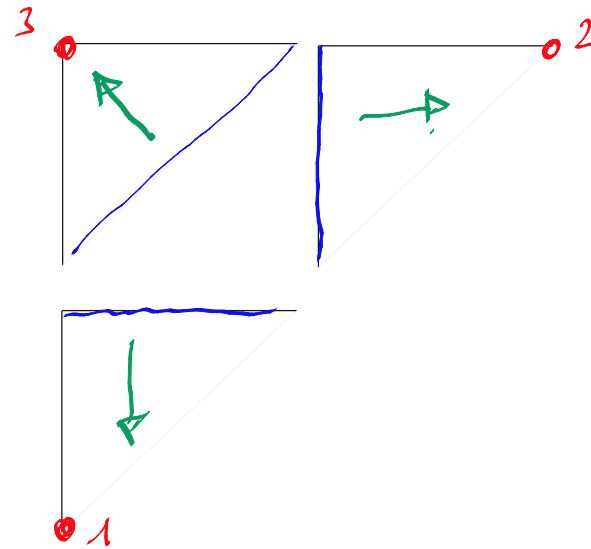
$$\sum \phi_{i,y} = 0$$



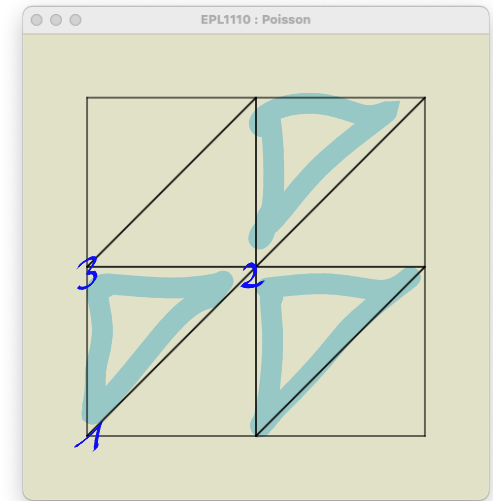
$$\frac{1}{2} \begin{pmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{pmatrix}$$

$$A_{ij}^e$$

Deux matrices locales



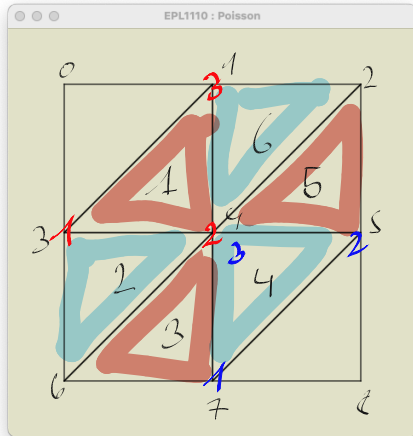
	$\phi_{i,x}$	$\phi_{i,y}$
$i = 1$	0	-1
2	-1	0
3	-1	1



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

A_{ij}^c

Assemblons les matrices



$$A_{ij}^e = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & -1 \\ -1 & -1 & 2 \end{bmatrix}$$

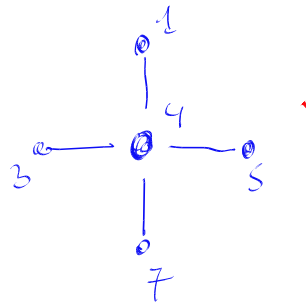
A_{ij}^e

$$A_{ij}^e = \begin{bmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{bmatrix}$$

A_{ij}^e

	1	2	3	4	5	6	7	8
1	1+2	-1		-1 -1				
2	-1	1+4			-1			
3			1+2	-1 -1		-1		
4	-1 -1		-1 -1	2+1+1 2+1+1	-1 -1		-1 -1	
5		-1		-1 -1	1+2			
6			-1			1+1 ...	-1	
7				-1 -1		-1	2+1	
8								

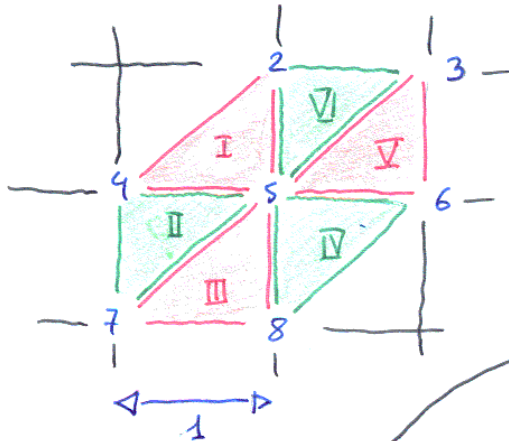
$$-2U_1 - 2U_3 + 8U_4 - 2U_5 - 2U_7$$



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

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



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

Triangle I (red) with nodes 1, 2, 3. Shape functions $\phi_{i,x}$ and $\phi_{i,y}$ are given by the matrix:

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

Resulting matrix:

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

Triangle II (green) with nodes 1, 2, 3. Shape functions $\phi_{i,x}$ and $\phi_{i,y}$ are given by the matrix:

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

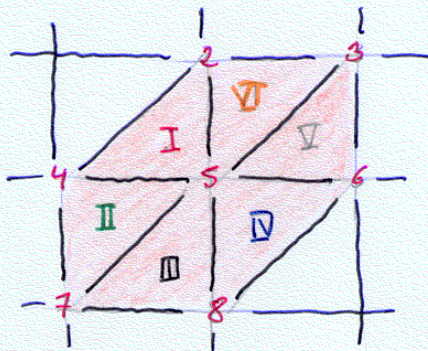
Resulting matrix:

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

CHECK !

$$\begin{aligned} \sum \phi_i &= 1 \\ \sum \phi_{i,x} &= 0 \\ \sum \phi_{i,y} &= 0 \end{aligned}$$

$$\begin{aligned} \phi_1 &= (1-x) + y \\ \phi_{1,x} &= -1 \\ \phi_{1,y} &= 0 \end{aligned}$$



ASSEMBLING LOCAL MATRICES

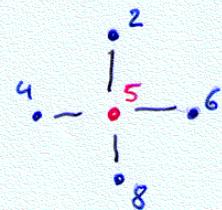
$$\begin{array}{c} \begin{array}{c} 3 \\ \diagup \quad \diagdown \\ 1 \quad 2 \end{array} \quad \begin{bmatrix} 1 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{bmatrix} \quad \frac{1}{2} \\ \begin{array}{c} 3 \\ \diagdown \quad \diagup \\ 1 \quad 2 \end{array} \quad \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & -1 \\ -1 & -1 & 2 \end{bmatrix} \quad \frac{1}{2} \end{array}$$

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

	2	3	4	5	6	7	8
2		12	-1		-1	-1	
3		-1	11			-1	
4				12	-1	-1	-1
5		-1	-1	211	211	-1	-1
6				-1	11	12	
7				-1		11	-1
8					-1	-1	21

$$\frac{1}{2} \begin{pmatrix} 8U_5 & -2U_2 & -2U_4 \\ -2U_8 & -2U_6 \end{pmatrix}$$



REGULAR
LATTICE
OF TURNER
TRIANGLES

=

CENTRAL
FINITE
DIFFERENCES

Une structure pour un système linéaire $Ax = b$

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
```

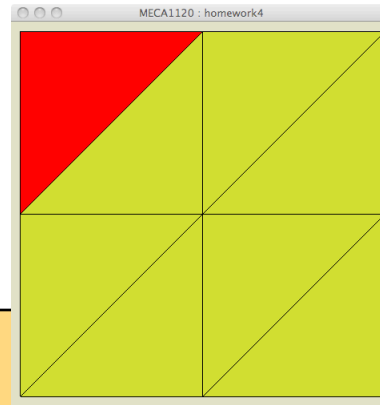
```
typedef struct {  
    double **A;  
    double *B;  
    int size;  
} femFullSystem;
```

```
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);
```

*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...

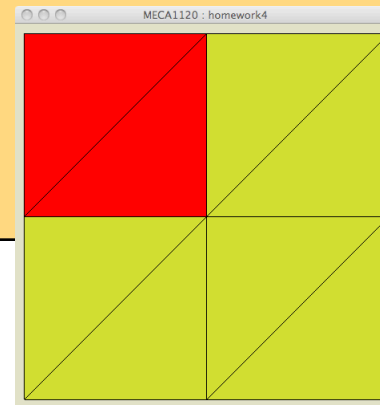


+1.0e+00	-5.0e-01	-5.0e-01
-5.0e-01	+5.0e-01	
-5.0e-01		+5.0e-01

:	+1.7e-01
:	+1.7e-01
:	+0.0e+00
:	+1.7e-01
:	+0.0e+00
:	+0.0e+00
:	+0.0e+00
:	+0.0e+00
:	+0.0e+00

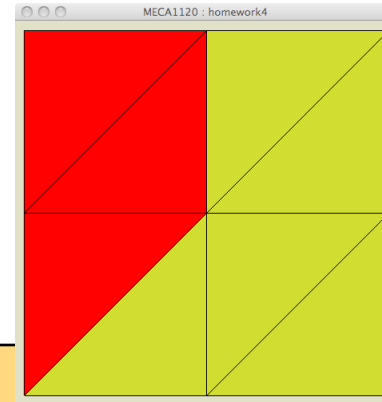
+1.0e+00	-5.0e-01	-5.0e-01	
-5.0e-01	+1.0e+00		-5.0e-01
-5.0e-01		+1.0e+00	-5.0e-01
-5.0e-01		-5.0e-01	+1.0e+00

:	+1.7e-01
:	+3.3e-01
:	+0.0e+00
:	+3.3e-01
:	+1.7e-01
:	+0.0e+00
:	+0.0e+00
:	+0.0e+00
:	+0.0e+00

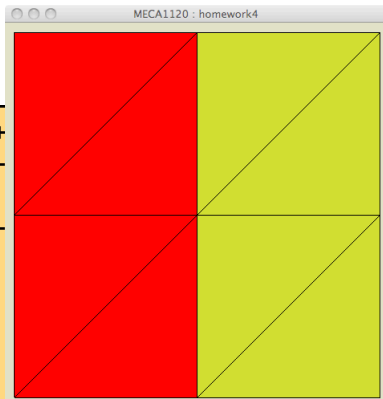


...par élément

Assemblage élément...



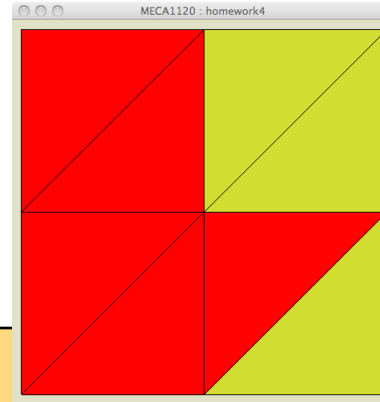
+1.0e+00	-5.0e-01	-5.0e-01		:	+1.7e-01
-5.0e-01	+1.0e+00		-5.0e-01	:	+3.3e-01
				:	+0.0e+00
-5.0e-01		+2.0e+00	-1.0e+00	:	+5.0e-01
	-5.0e-01	-1.0e+00	+1.5e+00	:	+3.3e-01
				:	+0.0e+00
		-5.0e-01		:	+1.7e-01
			+5.0e-01	:	+0.0e+00
				:	+0.0e+00



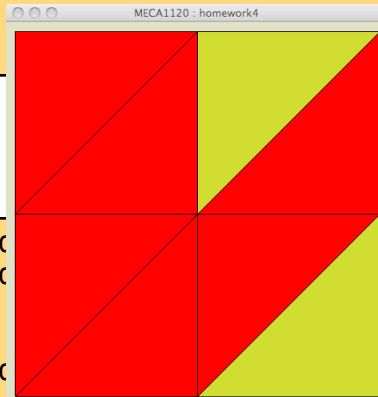
+		-5.0e-01		:	+1.7e-01
-			-5.0e-01	:	+3.3e-01
				:	+0.0e+00
-		+2.0e+00	-1.0e+00	:	+5.0e-01
		-1.0e+00	+2.0e+00	:	+5.0e-01
				:	+0.0e+00
		-5.0e-01		:	+3.3e-01
				:	+1.7e-01
				:	+0.0e+00

...par élément

Assemblage élément...



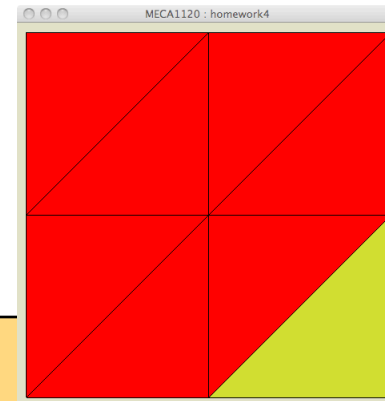
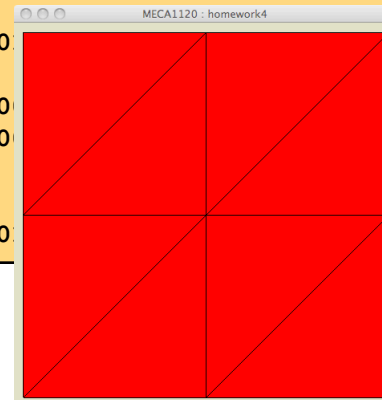
+1.0e+00	-5.0e-01	-5.0e-01				+1.7e-01
-5.0e-01	+1.0e+00		-5.0e-01			+3.3e-01
						: +0.0e+00
-5.0e-01		+2.0e+00	-1.0e+00	-5.0e-01		: +5.0e-01
	-5.0e-01	-1.0e+00	+3.0e+00	-5.0e-01	-1.0e+00	: +6.7e-01
			-5.0e-01	+5.0e-01		: +1.7e-01
		-5.0e-01			+1.0e+00	: +3.3e-01
					-5.0e-01	: +3.3e-01
					+1.5e+00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



+1.0e+00	-5.0e-01					: +1.7e-01
-5.0e-01	+1.0e+00					: +3.3e-01
						: +1.7e-01
-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	: +8.3e-01
			-5.0e-01	+5.0e-01		: +3.3e-01
		-5.0e-01			+1.0e+00	: +3.3e-01
					-5.0e-01	: +3.3e-01
					+1.5e+00	: +0.0e+00

...par élément

Assemblage élément...

[illegible]

```

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

```

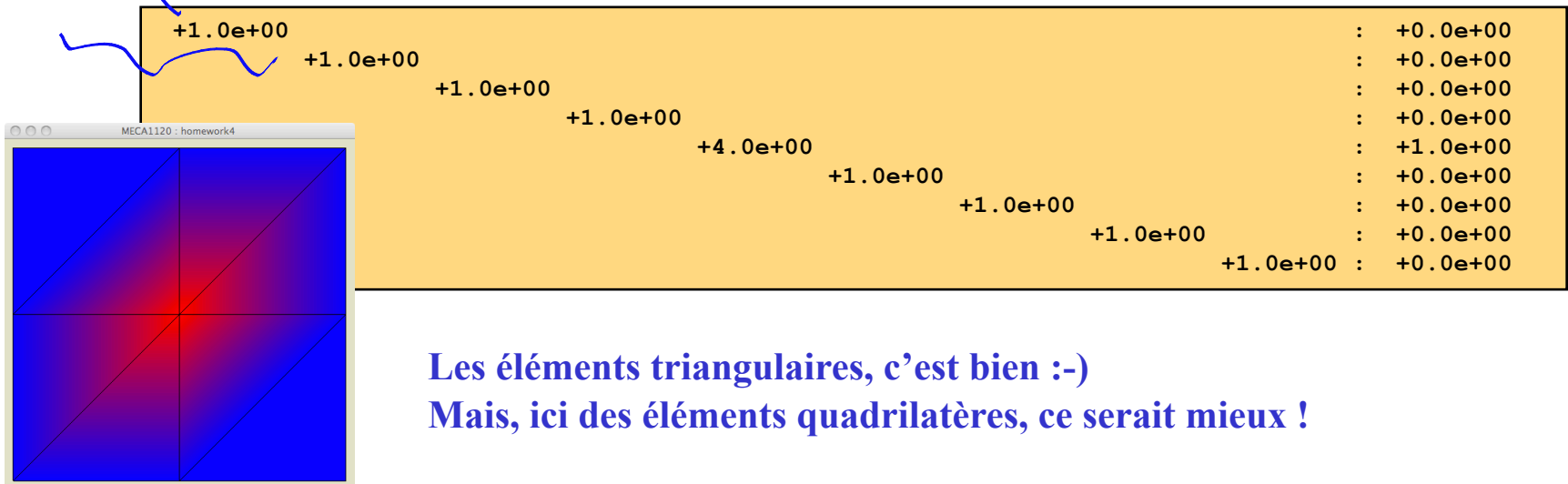
...par élément

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)
```

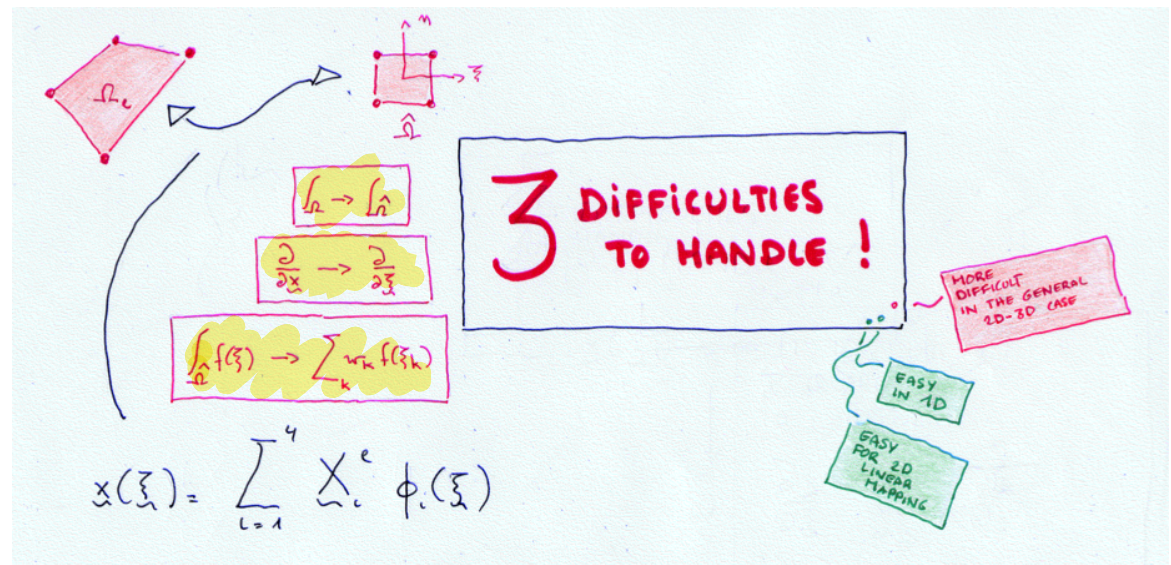
$U_0 = 0$
 $U_1 = 0$

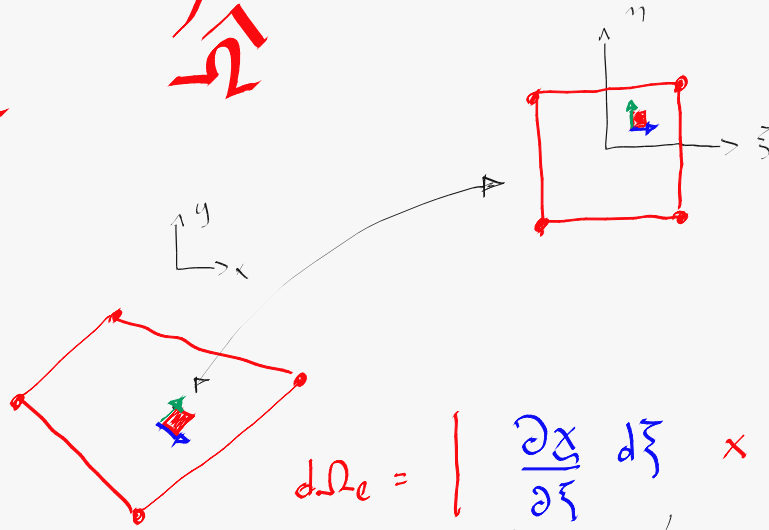
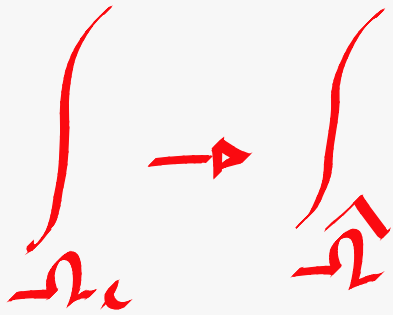


Et on fait
comment
pour des
quadrilatères ?

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

$$B_i = \int_{\Omega} f \tau_i d\Omega.$$





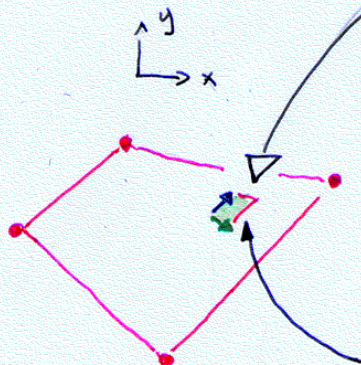
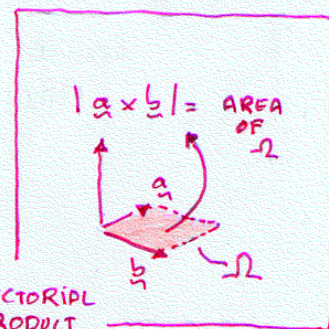
$$d\hat{\Omega} = d\xi d\eta = \begin{bmatrix} d\xi \\ 0 \end{bmatrix} \times \begin{bmatrix} 0 \\ d\eta \end{bmatrix}$$

$$d\Omega_c = \left| \underbrace{\frac{\partial \underline{x}}{\partial \xi}}_{\text{VECTOR}} d\xi \times \underbrace{\frac{\partial \underline{x}}{\partial \eta}}_{\text{VECTOR}} d\eta \right|$$

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

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

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



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

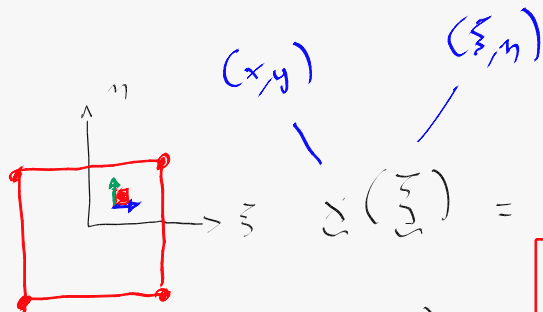
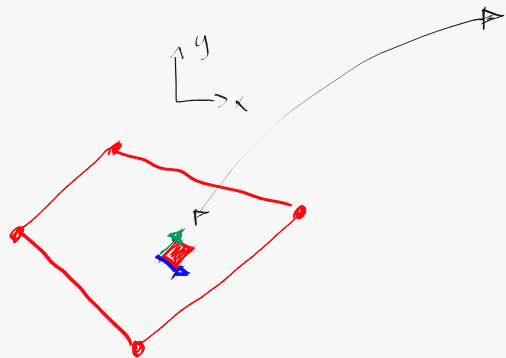
LENGTH

$$= \left| \frac{\partial x}{\partial \xi} \times \frac{\partial x}{\partial \eta} \right| d\xi d\eta$$

VECTORIAL PRODUCT

$$= \det \left(\frac{\partial x}{\partial \xi} \right) d\hat{\Omega}$$

J_e



(x, y) (ξ, η)

$$\xi(\xi) = \sum_{i=1}^4 X_i^e \phi_i(\xi)$$

$$d\xi(\xi) =$$

$$\sum_{i=1}^4 X_i^e \frac{\partial \phi_i(\xi)}{\partial \xi}$$

$$\frac{\partial \mathbf{x}}{\partial \xi}$$

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

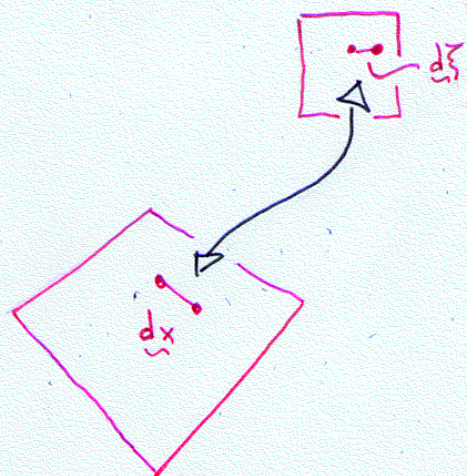
$d\xi$

$$u = \sum U_i^e \phi_i$$

$$x = \sum X_i^e \phi_i$$

$$y = \sum Y_i^e \phi_i$$

$$J_e = \det \left(\frac{\partial \mathbf{x}}{\partial \xi} \right)$$



$$x(\xi) = \sum_{i=1}^4 X_i^e \phi_i(\xi)$$

$$dx(\xi)$$

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



$$= \sum_{i=1}^4 X_i^e \frac{\partial \phi_i(\xi)}{\partial \xi} \cdot d\xi$$

$$\frac{\partial x}{\partial \xi}$$

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

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

How to
CALCULATE
 J^e ?

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

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

$$\xi(x)$$

PAS EVIDENT
A OBTENIR

$$x(\xi) = \sum x_i \phi_i(\xi)$$

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

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



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

$$\text{dog} = \left[\text{dog} \right]^{-1}$$

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

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

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

$\nabla \phi_i$

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

$$\frac{\partial \xi}{\partial x}$$

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

$$\frac{\partial x}{\partial \xi}$$



$$= \sum_{l=1}^4 X_l^e \frac{\partial \phi_i}{\partial \xi}$$



$$= \left(\text{character} \right)^{-1}$$

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

A_y^e

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



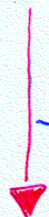
$$\frac{\partial \phi_i}{\partial x} = \frac{1}{J_e} \left(\underbrace{\frac{\partial y}{\partial \eta}}_{\sim} \underbrace{\frac{\partial \phi_i}{\partial \xi}}_{\sim} - \underbrace{\frac{\partial y}{\partial \xi}}_{\sim} \underbrace{\frac{\partial \phi_i}{\partial \eta}}_{\sim} \right)$$



$$\sum \gamma_i \phi_{i,\xi}$$

$$\sum \gamma_i \phi_{i,\eta}$$

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



$$\sum \gamma_i \frac{\partial \phi_i}{\partial \eta} \quad \frac{\partial \phi_i}{\partial x} = \frac{1}{J_c} \left(\frac{\partial \eta}{\partial \xi} \frac{\partial \phi_i}{\partial \xi} - \frac{\partial \eta}{\partial \xi} \frac{\partial \phi_i}{\partial \eta} \right) \quad \sum \gamma_i \frac{\partial \phi_i}{\partial \xi}$$

$J_c(\xi, \eta)$

$$A_y^e = \int_{\hat{\Omega}} \underbrace{\nabla \phi_i \cdot \nabla \phi_j}_{J_c(\xi, \eta)} J_c(\xi, \eta) d\hat{\Omega}$$

$J_c(\xi, \eta)$

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

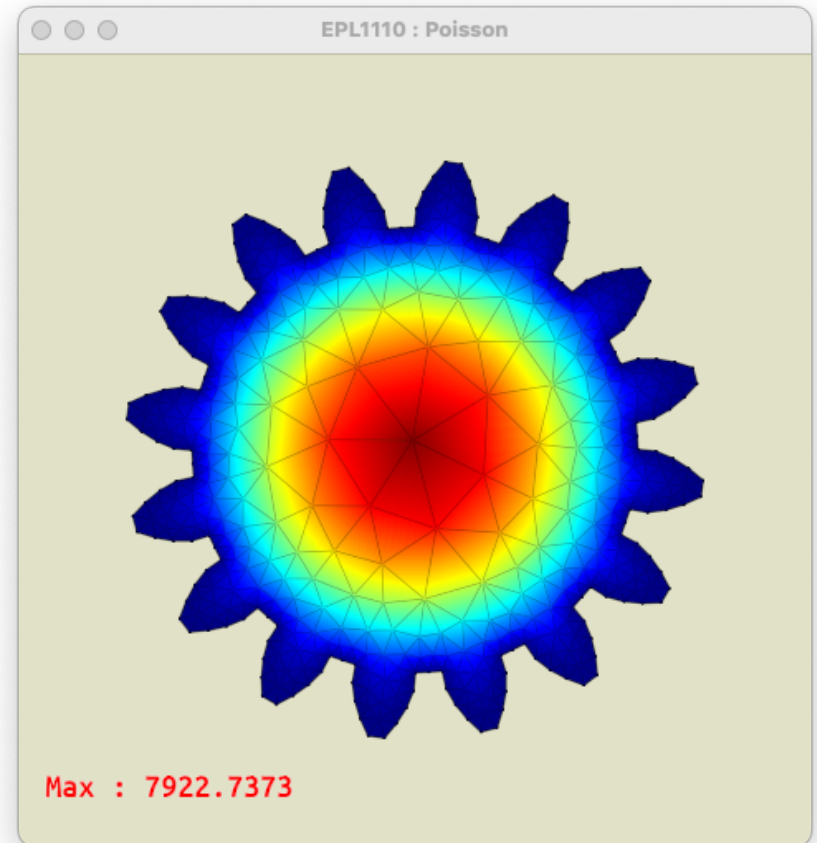
TO BE
SELECTED
IN A CLEVER WAY

$$\underbrace{\sum \phi_i \cdot \sum \phi_j}_{f(\xi)} \quad \underbrace{J_c(\xi)}_{f(\xi)}$$

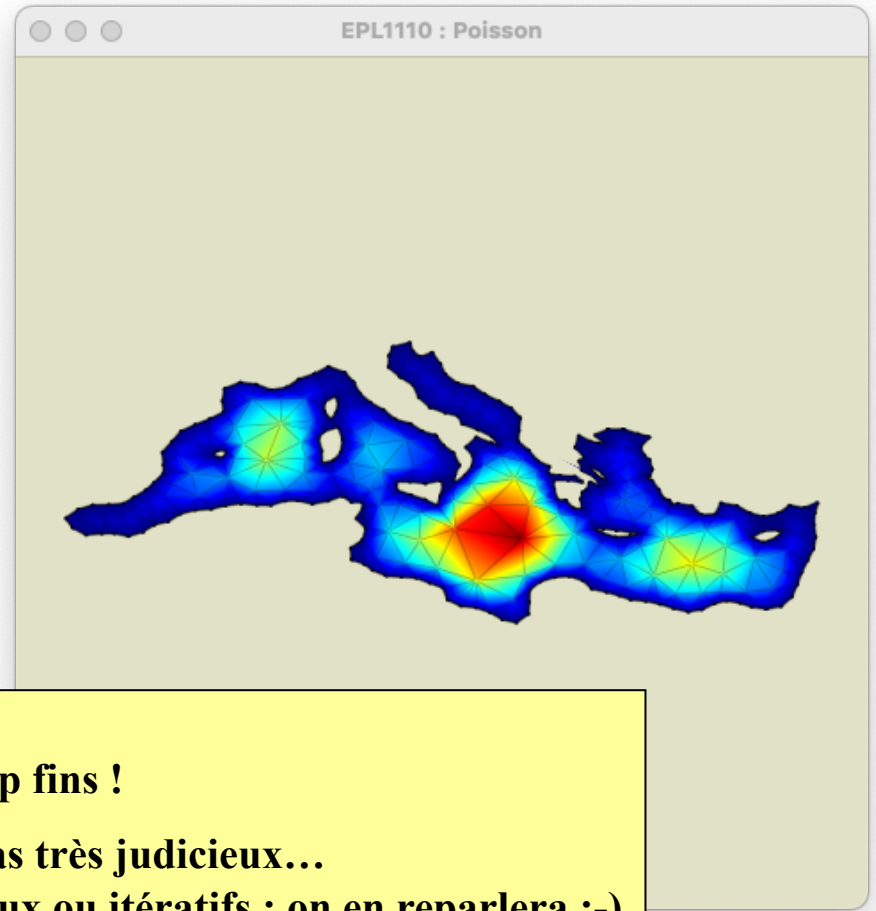
**GAUSS
LEGENDRE
QUADRATURE !**

Et zou
On est prêt !

$$\nabla^2 u + 1 = 0$$



**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...**