

## 1. Préliminaires

$$1) u_*(x, y) = \frac{(a-x) u_g(y) + (a+x) u_d(y)}{2a}$$

$u_*$  satisfait (2):

$$u_*(x, \pm b) = \frac{(a-x) u_g(\pm b) + (a+x) u_d(\pm b)}{2a} = 0$$

$u_*$  satisfait (3):

$$u_*(-a, y) = \frac{(a-(-a)) u_g(y) + (a+(-a)) u_d(y)}{2a} = \frac{2au_g(y)}{2a} = u_g(y)$$

$$u_*(a, y) = \frac{(a-a) u_g(y) + (a+a) u_d(y)}{2a} = \frac{2au_d(y)}{2a} = u_d(y)$$

De plus  $u_*$  est de classe  $C^2$  car somme de produit de fonctions de classe  $C^2$ .

$$2) w = u - u_*$$

$u$  et  $u_*$  respectent les conditions (2) et (3) donc sont égales au bord.

Donc  $w$  est nul au bord.

De plus  $u$  et  $u_*$  sont de classe  $C^2$  donc  $w$  est aussi de classe  $C^2$ .

Donc  $w \in V$ .

$$\forall v \in V \quad G = \int_{\Omega} \epsilon \nabla w \cdot \nabla v + \gamma \frac{\partial w}{\partial x} v + \lambda wv \, dx dy$$

$$G = \int_{\Omega} \epsilon \frac{\partial w}{\partial x} \frac{\partial v}{\partial x} \, dx dy + \int_{\Omega} \epsilon \frac{\partial w}{\partial y} \frac{\partial v}{\partial y} \, dx dy + \int_{\Omega} \gamma \frac{\partial w}{\partial x} v + \lambda wv \, dx dy$$

$$G = \int_{-b}^b \left( \int_{-a}^a \epsilon \frac{\partial w}{\partial x} \frac{\partial v}{\partial x} \, dx \right) dy + \int_{-b}^b \left( \int_{-a}^a \epsilon \frac{\partial w}{\partial y} \frac{\partial v}{\partial y} \, dy \right) dx + \int_{-b}^b \gamma \frac{\partial w}{\partial x} v + \lambda wv \, dx dy$$

$$G = \int_{-b}^b \left( \left[ \varepsilon \frac{\partial w}{\partial x} v \right]_{-a}^a - \int_{-a}^a \varepsilon \frac{\partial^2 w}{\partial x^2} v dx \right) dy + \int_{-a}^a \left( \left[ \varepsilon \frac{\partial w}{\partial y} v \right]_{-b}^b - \int_{-b}^b \varepsilon \frac{\partial^2 w}{\partial y^2} v dy \right) dx$$

$= 0$  car  $w$  nul sur bord.

$$+ \int_{\Omega} \gamma \frac{\partial w}{\partial x} v + \lambda w v dx dy$$

$$G = \int_{-b}^b \int_{-a}^a -\varepsilon \frac{\partial^2 w}{\partial x^2} v dx dy + \int_{-a}^a \int_{-b}^b -\varepsilon \frac{\partial^2 w}{\partial y^2} v dy dx + \int_{\Omega} \gamma \frac{\partial w}{\partial x} v + \lambda w v dx dy$$

$$G = \int_{\Omega} -\varepsilon \frac{\partial^2 w}{\partial x^2} v - \varepsilon \frac{\partial^2 w}{\partial y^2} v + \gamma \frac{\partial w}{\partial x} v + \lambda w v dx dy$$

$$G = \int_{\Omega} \left( -\varepsilon \frac{\partial^2 w}{\partial x^2} - \varepsilon \frac{\partial^2 w}{\partial y^2} + \gamma \frac{\partial w}{\partial x} + \lambda w \right) v dx dy$$

$$w = u - u_* \quad \text{or} \quad -\varepsilon \frac{\partial^2 u}{\partial x^2} - \varepsilon \frac{\partial^2 u}{\partial y^2} + \gamma \frac{\partial u}{\partial x} + \lambda u = 0$$

$$\text{Donc } G = \int_{\Omega} \left( \varepsilon \frac{\partial^2 u_*}{\partial x^2} + \varepsilon \frac{\partial^2 u_*}{\partial y^2} - \gamma \frac{\partial u_*}{\partial x} - \lambda u_* \right) v dx dy$$

$$\text{En posant } f_h = \varepsilon \frac{\partial^2 u_*}{\partial x^2} + \varepsilon \frac{\partial^2 u_*}{\partial y^2} - \gamma \frac{\partial u_*}{\partial x} - \lambda u_*$$

$$\text{On obtient } G = \int_{\Omega} f_h v dx dy.$$

4) Soient  $w_1$  et  $w_2$  deux solutions de (4).

$$\begin{aligned} \forall v \in V \quad a_h(w_1, v) = l_h(v) \\ a_h(w_2, v) = l_h(v) \end{aligned} \quad \Rightarrow \quad a_h(w_1, v) = a_h(w_2, v)$$

$$\Rightarrow a_h(w_1, v) - a_h(w_2, v) = 0$$

$$a_h \text{ est bilinéaire donc } a_h(w_1 - w_2, v) = 0 \quad \forall v \in V$$

$$w_1 - w_2 \in V \quad \text{donc} \quad a_h(w_1 - w_2, w_1 - w_2) = 0$$

$$\text{On montre que } \forall w \in V \quad \int_{\Omega} \frac{\partial w}{\partial x} w \, dx \, dy = 0$$

$$\int_{\Omega} \frac{\partial w}{\partial x} w \, dx \, dy = \int_{-b}^b \int_{-a}^a w \frac{\partial w}{\partial x} \, dx \, dy = \int_{-b}^b \left( \underbrace{[w]_{-a}^a}_{=0} - \int_{-a}^a \frac{\partial w}{\partial x} w \, dx \right) dy$$

$= 0$  car  $w$  n'est pas borné.

$$\int_{\Omega} \frac{\partial w}{\partial x} w \, dx \, dy = \int_{-b}^b \int_{-a}^a w \frac{\partial w}{\partial x} \, dx \, dy = - \int_{\Omega} \frac{\partial w}{\partial x} w \, dx \, dy$$

$$\Leftrightarrow 2 \int_{\Omega} \frac{\partial w}{\partial x} w \, dx \, dy = 0 \Leftrightarrow \int_{\Omega} \frac{\partial w}{\partial x} w \, dx \, dy = 0$$

$$\Rightarrow a_h(w, w) = \int_{\Omega} \varepsilon \left( \frac{\partial w}{\partial x} \right)^2 + \varepsilon \left( \frac{\partial w}{\partial y} \right)^2 + \lambda w^2 \, dx \, dy$$

$$\Rightarrow a_h(w_1 - w_2, w_1 - w_2) = \int_{\Omega} \underbrace{\varepsilon \left( \frac{\partial (w_1 - w_2)}{\partial x} \right)^2 + \varepsilon \left( \frac{\partial (w_1 - w_2)}{\partial y} \right)^2}_{\geq 0} + \lambda (w_1 - w_2)^2 \, dx \, dy$$

$$\text{Or } a_h(w_1 - w_2, w_1 - w_2) = 0 \quad \text{Donc } w_1 - w_2 = 0$$

Donc  $w_1 = w_2$ , ce qui prouve que (4) a au plus une solution.

5)  $u_\eta$  est de classe  $C^2$  sur le fermé  $\bar{\Omega}$ .  $u_\eta$  admet donc un minimum sur  $\bar{\Omega}$ .

Si ce minimum est atteint sur le bord, alors il est positif car  $u_\eta$  est positif au bord. Donc  $u_\eta \geq 0$  sur  $\bar{\Omega}$  donc sur  $\Omega$  aussi.

Sinon, ce minimum est atteint en un point  $p \in \Omega$ .

$$\text{Alors } \frac{\partial^2 u_\eta}{\partial x^2}(p) \geq 0, \quad \frac{\partial^2 u_\eta}{\partial y^2}(p) \geq 0, \quad \frac{\partial u_\eta}{\partial x} = 0$$

$$\text{Or } -\varepsilon \underbrace{\frac{\partial^2 u_\eta}{\partial x^2}(p)}_{\geq 0} - \varepsilon \underbrace{\frac{\partial^2 u_\eta}{\partial y^2}(p)}_{\geq 0} + \gamma \underbrace{\frac{\partial u_\eta}{\partial x}(p)}_{= 0} + \lambda u_\eta(p) = 0$$

$$\leq 0 \quad \geq 0$$

Donc  $u_\eta(p) > 0 \Rightarrow u_\eta \geq 0$  sur  $\Omega$ .

$$6) \text{ a) } \frac{\partial u}{\partial x}(x, y) + u(x, y) = 0 \Rightarrow u \text{ de la forme } f(y) e^{-x}$$

$$u(-1, y) = u_g(y) \Leftrightarrow f(y) e^{-1} = u_g(y) \Rightarrow f(y) = \frac{u_g(y)}{e}$$

$$\text{Donc il n'y a qu'une seule solution : } u_0(x, y) = \frac{u_g(y)}{e} e^{-x}$$

$$b) u_d(y) = u_0(1, y) = \frac{u_g(y)}{e} e^{-1} = \frac{u_g(y)}{e^2}$$

Le problème (1)-(2)-(3) admet une solution dans ce cas où  $u_d(y) = \frac{u_g(y)}{e^2}$ .

$$7) -\varepsilon \frac{\partial^2 u_\eta}{\partial x^2}(x,y) - \varepsilon \frac{\partial^2 u_\eta}{\partial y^2}(x,y) + \gamma \frac{\partial u_\eta}{\partial x}(x,y) + \lambda u_\eta(x,y) = 0$$

$$u_\eta(x,y) = U(x) \sin(\pi y)$$

$$\Rightarrow -\varepsilon U''(x) \sin(\pi y) + \varepsilon \pi^2 U(x) \sin(\pi y) + \gamma U'(x) \sin(\pi y) + \lambda U(x) \sin(\pi y) = 0$$

$$\Leftrightarrow \sin(\pi y) \left( -\varepsilon U''(x) + \varepsilon \pi^2 U(x) + \gamma U'(x) + \lambda U(x) \right) = 0$$

$$\Rightarrow -\varepsilon U''(x) + \gamma U'(x) + (\varepsilon \pi^2 + \lambda) U(x) = 0$$

$$\Leftrightarrow -U''(x) + \frac{\gamma}{\varepsilon} U'(x) + \left( \pi^2 + \frac{\lambda}{\varepsilon} \right) U(x) = 0$$

$$\text{On pose } A = \frac{1}{2} \left( \frac{\gamma}{\varepsilon} - \sqrt{\frac{\gamma^2}{\varepsilon^2} + 4\pi^2 + 4\frac{\lambda}{\varepsilon}} \right)$$

$$\text{et } B = \frac{1}{2} \left( \frac{\gamma}{\varepsilon} + \sqrt{\frac{\gamma^2}{\varepsilon^2} + 4\pi^2 + 4\frac{\lambda}{\varepsilon}} \right)$$

$$\Rightarrow U(x) = C_1 e^{Ax} + C_2 e^{Bx} \quad \text{avec } C_1 \text{ et } C_2 \text{ des constantes.}$$

$$u_\eta(-1, y) = \sin(\pi y) = U(-1) \sin(\pi y) \Rightarrow U(-1) = 1$$

$$u_\eta(1, y) = 0 = U(1) \sin(\pi y) \Rightarrow U(1) = 0$$

$$U(1) = 0 \Leftrightarrow C_1 e^A + C_2 e^B = 0 \Leftrightarrow C_1 e^A = -C_2 e^B \\ \Leftrightarrow C_1 = -C_2 e^{B-A}$$

$$U(-1) = 1 \Leftrightarrow C_1 e^{-A} + C_2 e^{-B} = 1$$

$$\Leftrightarrow -C_2 e^{B-A} e^{-A} + C_2 e^{-B} = 1 \Leftrightarrow C_2 (-e^{B-2A} + e^{-B}) = 1$$

$$\Leftrightarrow C_2 = \frac{1}{e^{-B} - e^{B-2A}}$$

$$C_1 = -C_2 e^{B-A} = \frac{-e^{B-A}}{e^{-B} - e^{B-2A}} = \frac{1}{-e^{A-2B} + e^{-A}}$$

$$C_1 = \frac{1}{e^{-A} - e^{A-2B}}$$

$$A = \frac{1}{2} \left( \frac{\gamma}{\epsilon} - \sqrt{\frac{\gamma^2}{\epsilon^2} + 4\pi^2 + 4 \frac{\lambda}{\epsilon}} \right)$$

$$B = \frac{1}{2} \left( \frac{\gamma}{\epsilon} + \sqrt{\frac{\gamma^2}{\epsilon^2} + 4\pi^2 + 4 \frac{\lambda}{\epsilon}} \right)$$

$$C_1 = \frac{1}{e^{-A} - e^{A-2B}}$$

$$C_2 = \frac{1}{e^{-B} - e^{B-2A}}$$

$$U(x) = C_1 e^{Ax} + C_2 e^{Bx}$$

$$u_\gamma^\rho(x, y) = (C_1 e^{Ax} + C_2 e^{Bx}) \sin(\pi y)$$

8) Soient  $U$  et  $V$  deux ouverts de  $\mathbb{R}^m$ ,  
 $\Phi : U \rightarrow V$  un  $C^1$ -diffeomorphisme  
et  $f$  une application de  $V$  dans  $\mathbb{R}$ .

Si  $U$  et  $V$  sont bornés et si  $f$  et  $(f \circ \Phi) \times |\det J_\Phi|$  sont Riemann-intégrables, alors

$$\int_V f = \int_U (f \circ \Phi) \times |\det J_\Phi|$$

## 2 - Le mariage

10) On prend  $N = 5$  et  $M = 4$

| 24 | 25 | 26 | 27 | 28 | 29 |    |    |    |    |
|----|----|----|----|----|----|----|----|----|----|
| 18 | 8  | 19 | 9  | 20 | 10 | 21 | 11 | 22 | 23 |
| 12 | 4  | 13 | 5  | 14 | 6  | 15 | 7  | 16 | 17 |
| 6  | 0  | 7  | 1  | 8  | 2  | 9  | 3  | 10 | 11 |
| 0  | 1  | 2  | 3  | 4  | 5  |    |    |    |    |

11) On considère la division euclidienne de  $s$  par  $N+1$ :  $s = (N+1)j' + i'$ ,  $0 \leq i' \leq N$ .

De plus,  $0 \leq s \leq (N+1)(M+1) + 1$  donc  $0 \leq j' \leq M$ .

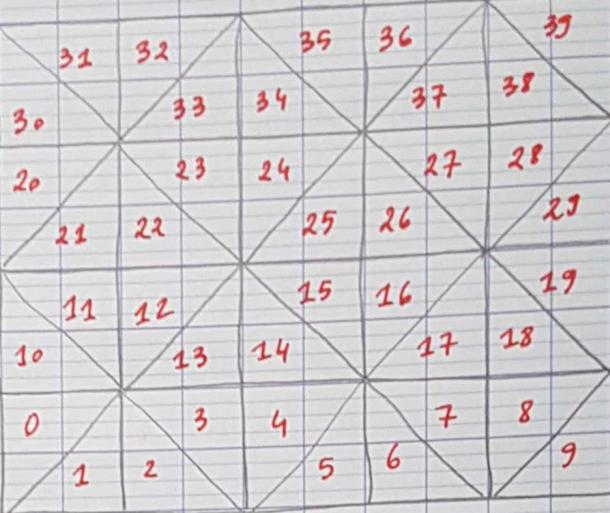
On a montré qu'il existe un unique couple  $(i', j')$  tel que  $0 \leq i' \leq N$ ,  $0 \leq j' \leq M$  et  $s = s_{i', j'}$ .

12) On considère la division euclidienne de  $k$  par  $N-1$ :  $k = (N-1)q + r$ ,  $0 \leq r \leq N-2$ .

De plus,  $0 \leq k \leq (N-1)(M-1) - 1$  donc  $0 \leq q \leq M-2$ .

On pose  $j = q + 1$  et  $i = r + 1$ . On a montré qu'il existe un unique couple  $(i', j')$  tel que  $1 \leq i' \leq N-1$ ,  $1 \leq j' \leq M-1$  et  $k = k_{i', j'}$ .

13) On reprend l'exemple de la question 10):



21) Supposons que  $f$  est affine, montrons que  
 $g(\vec{x}) = f(\vec{x}) - f(0)$  est linéaire. Soient  $\vec{x}, \vec{y} \in \mathbb{R}^n$   
et  $\alpha, \beta \in \mathbb{R}$ . On remarque :

$$\alpha \vec{x} + \beta \vec{y} = \alpha \vec{x} + (1-\alpha) \frac{\beta}{(1-\alpha)} \vec{y} = \alpha \vec{x} + (1-\alpha) \vec{z}$$

où  $\vec{z} = \gamma \vec{y} \in \mathbb{R}^n$ .

$$\begin{aligned} \text{Ainsi, } g(\alpha \vec{x} + \beta \vec{y}) &= f(\alpha \vec{x} + \beta \vec{y}) - f(0) \\ &= f(\alpha \vec{x} + (1-\alpha) \vec{z}) - f(0) \\ &= \alpha f(\vec{x}) + (1-\alpha) f(\vec{z}) - \alpha f(0) - \\ &\quad (1-\alpha) f(0) \\ &= \alpha g(\vec{x}) + (1-\alpha) g(\vec{y}) \end{aligned}$$

$$\begin{aligned} \text{Or, } (1-\alpha) g(\vec{z}) &= (1-\alpha) g(\gamma \vec{y}) \\ &= (1-\alpha) (f(\gamma \vec{y}) - f(0)) \\ &= (1-\alpha) (f(\gamma \vec{y}) + (1-\gamma) 0 - f(0)) \\ &= (1-\alpha) (\gamma f(\vec{y}) + (1-\gamma) f(0) - f(0)) \\ &= (1-\alpha) (\gamma g(\vec{y})) \\ &= \beta g(\vec{y}) \end{aligned}$$

$$\text{D'où } g(\alpha \vec{x} + \beta \vec{y}) = \alpha g(\vec{x}) + \beta g(\vec{y}).$$

Supposons que  $f$  est linéaire, montrons que  $f$

et affine. Soient  $\vec{x}, \vec{y} \in \mathbb{R}^n$  et  $\alpha \in \mathbb{R}$ :

$$\begin{aligned} f(\alpha \vec{x} + (1-\alpha) \vec{y}) &= f(\alpha \vec{x} + (1-\alpha) \vec{y}) + f(0) - f(0) \\ &= g(\alpha \vec{x} + (1-\alpha) \vec{y}) + f(0) \\ &= \alpha g(\vec{x}) + (1-\alpha) g(\vec{y}) + f(0) \\ &= \alpha(f(\vec{x}) - f(0)) + (1-\alpha)(f(\vec{y}) - f(0)) + f(0) \\ &= \alpha f(\vec{x}) + (1-\alpha) f(\vec{y}) \end{aligned}$$

22) Supposons que  $f$  est affine, montrons qu'elle est alors de la forme demandée. D'après 21)  $(x, y) \mapsto f(x, y) - f(0, 0)$  est linéaire donc  $\exists \alpha, \beta \in \mathbb{R}$  tels que  $f(x, y) - f(0, 0) = \alpha x + \beta y$ .  
D'où  $f(x, y) = \alpha x + \beta y + f(0, 0)$   
 $= \alpha x + \beta y + \gamma$ ,  $\gamma = f(0, 0) \in \mathbb{R}$ .

Supposons que  $f$  est de la forme demandée, montrons qu'elle est alors affine. On remarque que  $f(0, 0) = \gamma$ . Donc  $f(x, y) - f(0, 0) = \alpha x + \beta y$ . Ainsi  $(x, y) \mapsto f(x, y) - f(0, 0)$  est linéaire.  
D'après 21)  $f$  est affine.

$$23) \begin{pmatrix} x \\ y \end{pmatrix} = \sum_{j=0}^2 \pi_j \begin{pmatrix} x_j \\ y_j \end{pmatrix} \text{ et } \sum_{j=0}^2 \pi_j = 1.$$

Cela revient au système à 3 équations, 3 inconnues suivant :

$$\begin{cases} \pi_0 x_0 + \pi_1 x_1 + \pi_2 x_2 = x \\ \pi_0 y_0 + \pi_1 y_1 + \pi_2 y_2 = y \\ \pi_0 + \pi_1 + \pi_2 = 1 \end{cases}$$

où les  $\pi_i$ ,  $i \in \{0, 1, 2\}$  sont les inconnues. On peut réécrire le système sous forme matricielle :

$$\begin{pmatrix} x_0 & x_1 & x_2 \\ y_0 & y_1 & y_2 \\ 1 & 1 & 1 \end{pmatrix} \begin{pmatrix} \pi_0 \\ \pi_1 \\ \pi_2 \end{pmatrix} = \begin{pmatrix} x \\ y \\ \gamma \end{pmatrix}$$

$$A \cdot \pi = B$$

Il suffit de montrer que  $A$  est inversible pour montrer que le système possède une unique solution. Or, on sait que  $T$  n'est pas aplati donc  $\det A \neq 0$ .

24) On détermine l'inverse de  $A$ . On note  $d = \det A = x_0y_1 + x_1y_0 + x_2y_1 - (x_0y_1 + x_1y_0 + x_2y_1)$   
Donc, en calculant  $A^{-1}B$ , on a:

$$\pi_0: (x, y) \mapsto \frac{1}{d}((y_1 - y_2)x + (x_2 - x_1)y + (x_1y_2 - x_2y_1))$$

$$\pi_1: (x, y) \mapsto \frac{1}{d}((y_2 - y_0)x + (x_0 - x_2)y + (x_2y_0 - x_0y_2))$$

$$\pi_2: (x, y) \mapsto \frac{1}{d}((y_0 - y_1)x + (x_1 - x_0)y + (x_0y_1 - x_1y_0))$$

En utilisant 22), on a que les fonctions  $\pi_i$ ,  $i \in \{0, 1, 2\}$  sont affines.

25) En utilisant 24), on obtient:

$$\pi_j(x_i, y_i) = 0 \text{ pour } i \neq j.$$

26) On sait que  $f$  est affine de  $\mathbb{R}^2$  dans  $\mathbb{R}$  donc d'après 22)  $f$  est de la forme  $f(x, y) = \alpha x + \beta y + \gamma$ ,  $\alpha, \beta, \gamma \in \mathbb{R}$ ,  $\forall x, y \in \mathbb{R}$ .

$$\text{Donc, } f(x, y) = \alpha \left( \sum_{i=0}^2 \pi_i x_i \right) + \beta \left( \sum_{i=0}^2 \pi_i y_i \right) + \gamma$$

$$= \sum_{i=0}^2 \pi_i (\alpha x_i + \beta y_i) + \sum_{i=0}^2 \pi_i \gamma$$

$$= \sum_{i=0}^2 \pi_i (\alpha x_i + \beta y_i + \gamma)$$

$$= \sum_{i=0}^2 \pi_i f(x_i, y_i)$$

Ainsi, si  $x_i \in [0, 2]$ ,  $f(x_i, y_i) = 0$ , on a  $f(x, y) = 0$ .

27) En utilisant 24) avec les coordonnées  $(\hat{x}_0, \hat{y}_0)$ ,  $(\hat{x}_1, \hat{y}_1)$  et  $(\hat{x}_2, \hat{y}_2)$ . On a :

$$\hat{x}_0(x, y) = 1 - x - y$$

$$\hat{x}_1(x, y) = x$$

$$\hat{x}_2(x, y) = y$$

28) On a que  $F_T$  est affine donc d'après 21),  $\tilde{F}_T(x, y) = F_T(x, y) - F_T(0, 0)$  est linéaire.

Donc, il existe une matrice  $C$  à coefficients réels telle que :  $\tilde{F}_T(x, y) = C \begin{pmatrix} x \\ y \end{pmatrix}$ .

Déterminons  $C$ :

$$\tilde{F}_T(\hat{x}_1, \hat{y}_1) = F_T(1, 0) - F_T(0, 0) = \begin{pmatrix} x_1 - x_0 \\ y_1 - y_0 \end{pmatrix}$$

$$\tilde{F}_T(\hat{x}_2, \hat{y}_2) = F_T(0, 1) - F_T(0, 0) = \begin{pmatrix} x_2 - x_0 \\ y_2 - y_0 \end{pmatrix}$$

On en déduit :

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} x_1 - x_0 \\ y_1 - y_0 \end{pmatrix}$$

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} x_2 - x_0 \\ y_2 - y_0 \end{pmatrix}$$

D'où : 
$$\begin{cases} a = x_1 - x_0 \\ b = x_2 - x_0 \\ c = y_1 - y_0 \\ d = y_2 - y_0 \end{cases}$$

$$\Rightarrow C = \begin{pmatrix} x_1 - x_0 & x_2 - x_0 \\ y_1 - y_0 & y_2 - y_0 \end{pmatrix}$$

D'où  $F_T(x, y) = \begin{pmatrix} x_1 - x_0 & x_2 - x_0 \\ y_1 - y_0 & y_2 - y_0 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} x_0 \\ y_0 \end{pmatrix}$ .

29) Pour montrer que  $F_T$  est inversible, on donne son inverse :

$$F_T^{-1}(x, y) = C^{-1} \left( \begin{pmatrix} x \\ y \end{pmatrix} - \begin{pmatrix} x_0 \\ y_0 \end{pmatrix} \right).$$

30) On détermine la jacobienne de  $F_T$  :

$$B_T = \begin{pmatrix} x_1 - x_0 & x_2 - x_0 \\ y_1 - y_0 & y_2 - y_0 \end{pmatrix}$$

$$32) \det B_T = (x_2 - x_0)(y_2 - y_0) - (x_1 - x_0)(y_1 - y_0)$$

Déterminons l'aire de  $T$  que l'on note  $A_T$  :

$$A_T = \frac{1}{2} (x_0(y_1 - y_2) + x_1(y_2 - y_0) + x_2(y_0 - y_1))$$

$$= \frac{1}{2} (x_0 y_1 - x_0 y_2 + x_1 y_2 - x_1 y_0 + x_2 y_0 - x_2 y_1)$$

$$= \frac{1}{2} ((x_1 - x_0)(y_2 - y_0) - (x_2 - x_0)(y_1 - y_0))$$

$$= \frac{1}{2} \det B_T$$

33) Soit  $(\bar{x}, \bar{y}) \in \mathbb{R}^2$ , on note :  $(\hat{x}, \hat{y}) = F_T^{-1}(\bar{x}, \bar{y})$   
On a  $(\hat{x}, \hat{y}) = \sum_{i=0}^2 \lambda_i (\hat{x}_i, \hat{y}_i)$

$$\begin{aligned} \text{Donc, comme } F_T \text{ est affine et que } \sum_{i=0}^2 \lambda_i = 1, \\ \text{on a : } (\bar{x}, \bar{y}) &= F_T(\hat{x}, \hat{y}) \\ &= \sum_{i=0}^2 \lambda_i (\hat{x}, \hat{y}) F_T(\hat{x}_i, \hat{y}_i) \\ &= \sum_{i=0}^2 \lambda_i (\hat{x}, \hat{y}) (x_i, y_i) \end{aligned}$$

Donc les  $\lambda_i (\hat{x}, \hat{y})$  respectent le système vu en 23). On en déduit que  $\lambda_i \in \{0, 1\}$   
 $\lambda_i (\hat{x}, \hat{y}) = \lambda_i (\bar{x}, \bar{y})$

Ainsi,  $\hat{\pi}_i(x, y) = \hat{\pi}_i(F_T^{-1}(x, y))$  où  $\hat{\pi}_i = \hat{\pi}_i \circ F_T^{-1}$ ,  $\forall i \in \{0, 1\}$

34) On obtient:  $\nabla \hat{\pi}_i = {}^t(B_T^{-1}) \cdot \nabla \hat{\pi}_i$ ,  $\forall i \in \{0, 1\}$ .

$\nabla \hat{\pi}_i$  est constant.

$$35) \bullet \int_T \nabla \hat{\pi}_k \nabla \hat{\pi}_j dx dy = {}^t(\nabla \hat{\pi}_k \nabla \hat{\pi}_j) \int_T dx dy =$$

$$= \nabla \hat{\pi}_k \nabla \hat{\pi}_j A_T = \frac{1}{2} {}^t(B_T^{-1}) \nabla \hat{\pi}_k (B_T^{-1}) \nabla \hat{\pi}_j \det(B_T)$$

$$\bullet \int_T \frac{\partial \hat{\pi}_k}{\partial x} \hat{\pi}_j dx dy = \int_T \frac{\partial \hat{\pi}_k}{\partial x} (F_T(x, y)) \hat{\pi}_j(F_T(x, y))$$

$$| \det B_T | dx dy = \int_T \frac{\partial \hat{\pi}_k}{\partial \hat{x}} (\hat{x}, \hat{y}) \hat{\pi}_j(\hat{x}, \hat{y})$$

$$| \det(B_T) | d\hat{x} d\hat{y} = (1, 0) \nabla \hat{\pi}_k | \det(B_T) | \int_T \hat{\pi}_j d\hat{x} d\hat{y}$$

$$\bullet \int_T \hat{\pi}_k \hat{\pi}_j dx dy = \int_T \hat{\pi}_k(F_T(x, y)) \hat{\pi}_j(F_T(x, y))$$

$$dx dy = | \det(B_T) | \int_T \hat{\pi}_k \hat{\pi}_j d\hat{x} d\hat{y}$$

36) Avec  $j = 0$ :

$$\int_T \hat{\pi}_j d\hat{x} d\hat{y} = \int_T \hat{\pi}_0 d\hat{x} d\hat{y}$$

$$= \int_0^1 \int_0^{1-\hat{x}} 1 - \hat{x} - \hat{y} d\hat{y} d\hat{x}$$

$$= \int_0^1 \left[ \hat{y} (1 - \hat{x}) - \frac{1}{2} \hat{y}^2 \right]_0^{1-\hat{x}} d\hat{x}$$

$$= \int_0^1 (1 - \hat{x})^2 - \frac{1}{2} (1 - \hat{x})^2 d\hat{x}$$

$$= \frac{1}{2} \left[ \frac{1}{3} \hat{x}^3 - \hat{x}^2 + \hat{x} \right]_0^1$$

$$= \frac{1}{6}$$

En appliquant le même procédé pour  $j=1$  et  $j=2$ , on trouve  $\int_T \hat{x}_j dx dy = \frac{1}{6}$  pour tout  $j \in [0, 2]$

Et  $\int_T \hat{x}_k \hat{x}_j dx dy = \begin{cases} \frac{1}{24} & \text{si } k+j \\ \frac{1}{12} & \text{sinon} \end{cases}$

### 3 - Le problème discret

38) On remarque que les éléments de  $V_h$  sont caractérisés par leurs valeurs aux nœuds intérieurs. On a donc que  $I_h$  est un isomorphisme. D'où claim  $V_h = I$ .

39) On remarque que l'élément  $w_k$ ,  $k \in [0, I-1]$  valent 1 au nœud intérieur  $k$  et 0 ailleurs. On peut donc écrire tout élément  $v \in V_h$  comme une combinaison linéaire de  $w_k$ ,  $k \in [0, I-1]$ . Par ailleurs, on voit aisément que la famille  $(w_k)_{k \in [0, I-1]}$  est libre. Elle forme donc une base de  $V_h$ .

40) a) On a  $w_k|_T = 0$   
 b) On a  $w_k|_T = \chi^T_{\epsilon \{0, 1, 2\}}$   
 c) On a  $w_k = \chi^T_{i \in \{0, 1, 2\}}$  donc  $\nabla w_k = \nabla \chi^T = {}^t B_T^{-1} \cdot \nabla \chi$   
 Donc  $\nabla w_k \nabla w_j = {}^t B_T^{-1} \nabla \chi^T \cdot {}^t B_T^{-1} \nabla \chi_j$ .

41) On cherche  $w_\eta^h \in V_h$  tel que  $\forall v_h \in V_h$   $a_\eta(w_\eta^h, v_h) = l_\eta(v_h)$ . Or, on connaît une base de  $V_h$  (question 39)). Donc  $w_\eta^h = \sum \alpha_l w_l$  où  $\alpha_l \in \mathbb{R}$ ,  $\forall l \in [0, I-1]$ .

Alors, déterminer  $w_\eta^h$  revient à déterminer  $(\alpha_0, \alpha_1, \dots, \alpha_{I-1})$  tel que  $\sum_{l=0}^{I-1} \alpha_l a(w_l, w_k) = l_\eta(w_k)$  pour tout  $k \in [0, I-1]$ . Matriciellement :

$$\begin{pmatrix} a_\eta(w_0, w_0) & \cdots & a_\eta(w_{I-1}, w_0) \\ \vdots & \ddots & \vdots \\ a_\eta(w_0, w_{I-1}) & \cdots & a_\eta(w_{I-1}, w_{I-1}) \end{pmatrix} \begin{pmatrix} \alpha_0 \\ \vdots \\ \alpha_{I-1} \end{pmatrix} = \begin{pmatrix} l_\eta(w_0) \\ \vdots \\ l_\eta(w_{I-1}) \end{pmatrix}$$

$$A_\eta \quad U = B_\eta$$

45) a) Pour  $d=0$ :  $(x, y) \mapsto 1$

$$\cdot \int_T h(\hat{x}, \hat{y}) d\hat{x} d\hat{y} = \int_T d\hat{x} d\hat{y} = A_T = \frac{1}{2} -$$

$$\cdot \frac{1}{6} \times 1 + \frac{1}{6} \times 1 + \frac{1}{6} \times 1 = \frac{3}{6} = \frac{1}{2}.$$

Pour  $d=1$ :  $\begin{cases} (x, y) \mapsto x \\ (x, y) \mapsto y \end{cases}$

$$\cdot \int_T h(\hat{x}, \hat{y}) d\hat{x} d\hat{y} = \int_T \hat{x} d\hat{x} d\hat{y} = \int_0^1 \int_0^{1-\hat{x}} \hat{x} dy d\hat{x}$$

$$= \int_0^1 [\hat{y} \hat{x}]_0^{1-\hat{x}} d\hat{x} = \int_0^1 (1-\hat{x}) \hat{x} d\hat{x} = \left[ \frac{1}{2} \hat{x}^2 - \frac{1}{3} \hat{x}^3 \right]_0^1 = \frac{1}{2} - \frac{1}{3} = \frac{1}{6}.$$

$$\cdot \frac{1}{6} \times \frac{1}{2} + \frac{1}{6} \times 0 + \frac{1}{6} \times \frac{1}{2} = \frac{1}{6}.$$

Le raisonnement est identique pour  $(x, y) \mapsto y$ .

Pour  $d=2$ :  $\begin{cases} (x, y) \mapsto x^2 \\ (x, y) \mapsto y^2 \\ (x, y) \mapsto xy \end{cases}$

$$\cdot \int_T h(\hat{x}, \hat{y}) d\hat{x} d\hat{y} = \int_T x^2 d\hat{x} d\hat{y} = \int_T \hat{y} d\hat{x} d\hat{y} =$$

$$= \frac{1}{12}$$

$$\cdot \frac{1}{6} \left( \frac{1}{2} \right)^2 + \frac{1}{6} \left( \frac{1}{2} \right)^2 = \frac{1}{12}.$$

$$\cdot \int_T h(\hat{x}, \hat{y}) d\hat{x} d\hat{y} = \int_T xy d\hat{x} d\hat{y} = \frac{1}{24}$$

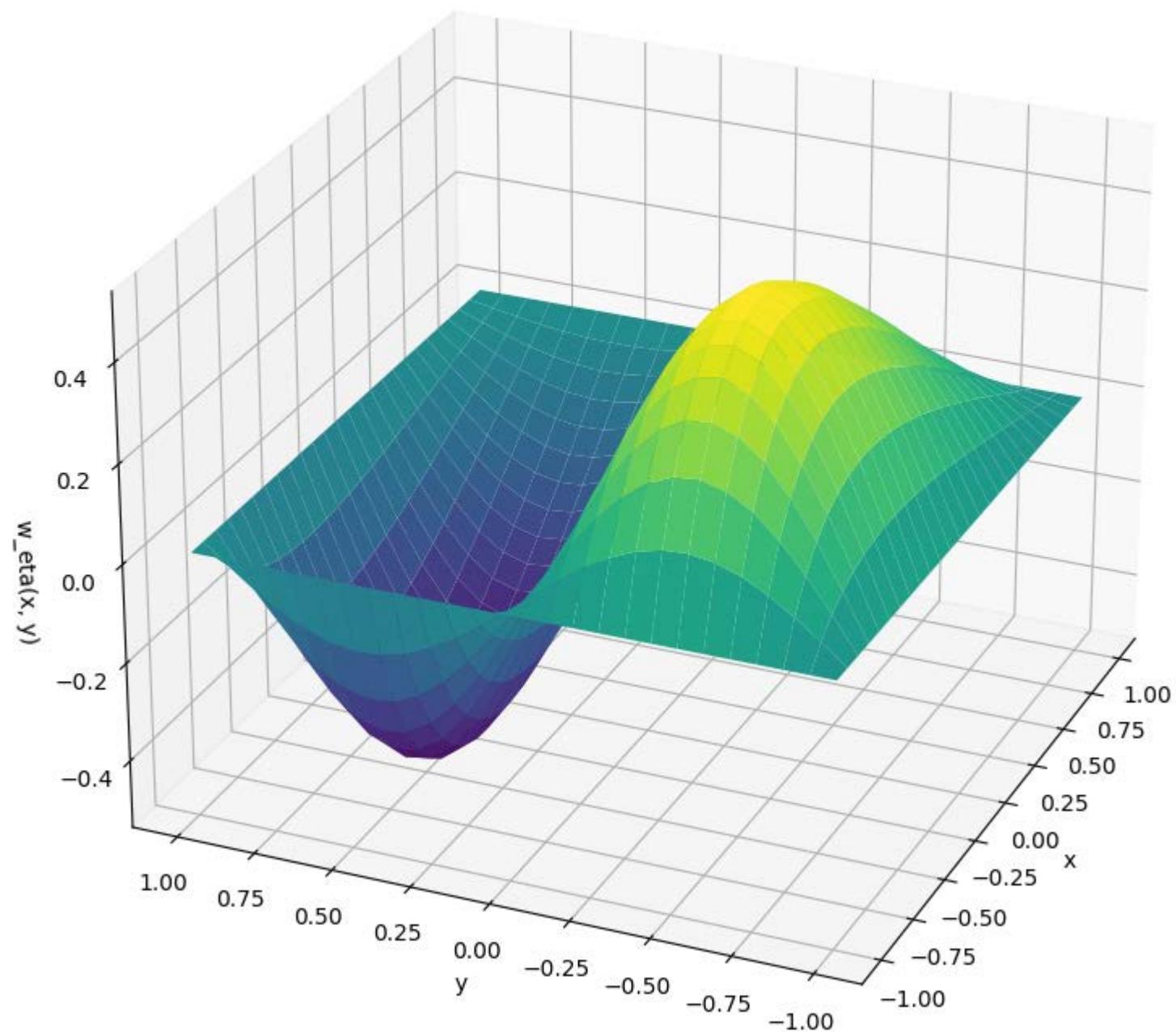
$$\cdot \frac{1}{6} \times 0 + \frac{1}{6} \times 0 + \frac{1}{6} \times \frac{1}{4} = \frac{1}{24}.$$

Pour  $d=3$  :  $(x, y) \mapsto x^3$

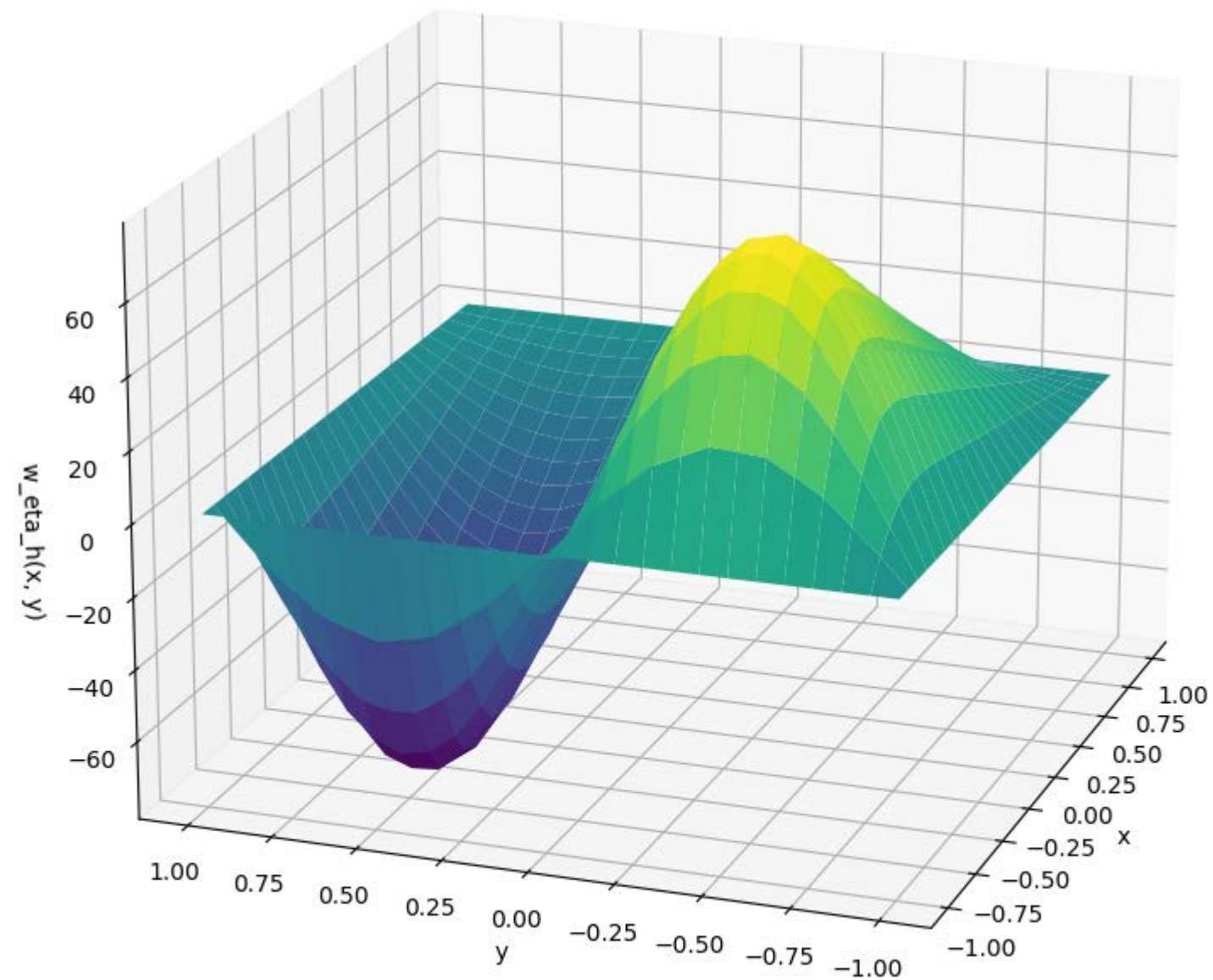
$$\begin{aligned} & \cdot \int_T h(\hat{x}, \hat{y}) d\hat{x} d\hat{y} = \int_T \hat{x}^3 d\hat{x} d\hat{y} = \frac{1}{20} \\ & \cdot \frac{1}{6} \left(\frac{1}{2}\right)^3 + \frac{1}{6} \left(\frac{1}{2}\right)^3 = \frac{1}{24} \end{aligned}$$

Donc la formule est vraie jusqu'à  $d=2$ .

solution exacte, N, M = 25



solution approchee, N, M = 25



# Projet

Corentin Marcou      Walid Abed

19 février 2023

## Table des matières

|          |                                    |           |
|----------|------------------------------------|-----------|
| <b>1</b> | <b>Variables globales</b>          | <b>1</b>  |
| <b>2</b> | <b>Point d'entrée du programme</b> | <b>2</b>  |
| <b>3</b> | <b>Classe Noeud</b>                | <b>4</b>  |
| 3.1      | header . . . . .                   | 4         |
| 3.2      | implémentation . . . . .           | 4         |
| <b>4</b> | <b>Classe Triangle</b>             | <b>5</b>  |
| 4.1      | header . . . . .                   | 5         |
| 4.2      | implémentation . . . . .           | 6         |
| <b>5</b> | <b>Classe Maillage</b>             | <b>7</b>  |
| 5.1      | header . . . . .                   | 7         |
| 5.2      | implémentation . . . . .           | 9         |
| <b>6</b> | <b>Boite à outils</b>              | <b>11</b> |
| 6.1      | header . . . . .                   | 11        |
| 6.2      | implémentation . . . . .           | 13        |

## 1 Variables globales

```
#ifndef DONNEES_DU_PROBLEME
#define DONNEES_DU_PROBLEME

extern double epsilon;
extern double gama;
extern double lambda;

extern double a;
extern double b;
extern int N;
extern int M;
```

```
extern double det;
```

```
#endif
```

## 2 Point d'entrée du programme

```
#define _USE_MATH_DEFINES
#include <cmath>
#include <fstream>
#include <iostream>
#include <vector>

#include "boite_a_outils.h"
#include "donnees_du_probleme.h"
#include "maillage.h"
#include "noeud.h"
#include "triangle.h"

using namespace std;

double epsilon = 1;
double gama = 1;
double lambda = 1;
double a = 1;
double b = 1;
int N = 12;
int M = 12;
double det = abs(2 * (a / N) * (b / M));

double u_g(double y) { return sin(M_PI * y); }
double u_gpp(double y) { return -M_PI * M_PI * sin(M_PI * y); }

double u_d(double y) { return 0; }
double u_dpp(double y) { return 0; }

double f_eta(double x, double y) {
    return f_second_membre(u_g, u_d, u_gpp, u_dpp, x, y);
}

int main(void) {
    int I = (N - 1) * (M - 1);
    Maillage maille;
    vector<double> B_eta = scd_membre(f_eta, maille);
    cout << "Le second membre est:" << endl;
```

```

for (int k = 0; k < I && k < 30; k++)
    cout << B_eta[k] << endl;
// w_eta_h est la solution approchée.
vector<double> w_eta_h = inv_syst(B_eta, maille, 5);
cout << endl << "La solution approchée est:" << endl;
for (int k = 0; k < I && k < 30; k++)
    cout << w_eta_h[k] << endl;
cout << endl << "Les erreurs sont:" << endl;
vector<double> erreur = erreurs(u_eta, w_eta_h, maille);
for (double err : erreur)
    cout << err << endl;
vector<double> solution_exacte;
for (int k = 0; k < I; k++) {
    vector<double> xy = maille.int_coord(k);
    solution_exacte.push_back(u_eta(xy[0], xy[1]));
}
vector<double> ecart = solution_exacte - w_eta_h;
cout << endl << "sol exacte\tsol approchée\tecart" << endl;
for (int k = 0; k < I && k < 30; k++)
    cout << solution_exacte[k] << "\t" << w_eta_h[k] << "\t" << ecart[k]
        << endl;
cout << endl;

// Ecriture des fichiers:
ofstream file;

// Ecriture des solutions exactes:
file.open("solution_exacte.txt");
for (double d : solution_exacte)
    file << d << endl;
file.close();

// Ecriture des solutions approchées:
file.open("solution_approchée.txt");
for (double d : w_eta_h)
    file << d << endl;
file.close();

vector<double> X0(B_eta.size(), 1);
vector<double> AX0 = mat_vec(X0, maille);
vector<double> AB_eta = mat_vec(B_eta, maille);
cout << "A * X0 \tA * B_eta" << endl;
for (int k = 0; k < I && k < 30; k++)
    cout << AX0[k] << "\t" << AB_eta[k] << endl;
cout << endl;

```

```
}
```

### 3 Classe Noeud

#### 3.1 header

```
#ifndef NOEUD_H
#define NOEUD_H

class Noeud {
    // Coordonnées du noeud
    double x, y;

public:
    // Constructeur par défaut.
    Noeud(void);
    // Constructeur.
    Noeud(double x, double y);

    // Getters.
    double get_x(void);
    double get_y(void);
};

#endif
```

#### 3.2 implémentation

```
#include "noeud.h"

Noeud::Noeud(void) {}

Noeud::Noeud(double x, double y) {
    this->x = x;
    this->y = y;
}

double Noeud::get_x(void) { return x; }

double Noeud::get_y(void) { return y; }
```

## 4 Classe Triangle

### 4.1 header

```
#ifndef TRIANGLE_H
#define TRIANGLE_H

#include <vector>

#include "donnees_du_probleme.h"
#include "noeud.h"

using namespace std;

class Triangle {
    // sommets du triangle.
    vector<Noeud> noeuds;

public:
    // Constructeur par défaut.
    Triangle(void);
    // Constructeurs.
    Triangle(Noeud n0, Noeud n1, Noeud n2);
    Triangle(vector<Noeud> noeuds);

    // Getter.
    vector<Noeud> get_noeuds(void);

    // Question 31:
    // Retourne la matrice B_T du triangle.
    vector<vector<double>> calc_mat_BT(void);

    // Retourne la matrice B_T du triangle avec une permutation des sommets.
    vector<vector<double>> calc_mat_BT(vector<int> permut);

    // Retourne l'inverse de la matrice B_T.
    vector<vector<double>> inv_mat_BT(void);

    // Question 37:
    vector<vector<double>> diff_terme(void);
    vector<vector<double>> convect_terme(void);
    vector<vector<double>> react_terme(void);
};

#endif
```

## 4.2 implémentation

```
#include <cmath>
#include <iostream>
#include <vector>

#include "triangle.h"

using namespace std;

Triangle::Triangle(void) {}

Triangle::Triangle(Noeud n0, Noeud n1, Noeud n2) {
    this->noeuds = {n0, n1, n2};
}

Triangle::Triangle(vector<Noeud> noeuds) { this->noeuds = noeuds; }

vector<Noeud> Triangle::get_noeuds(void) { return noeuds; }

vector<vector<double>> Triangle::calc_mat_BT() {
    return {{noeuds[1].get_x() - noeuds[0].get_x(),
              noeuds[2].get_x() - noeuds[0].get_x()},
            {noeuds[1].get_y() - noeuds[0].get_y(),
              noeuds[2].get_y() - noeuds[0].get_y()}};
}

vector<vector<double>> Triangle::calc_mat_BT(vector<int> permut) {
    return {{noeuds[permut[1]].get_x() - noeuds[permut[0]].get_x(),
              noeuds[permut[2]].get_x() - noeuds[permut[0]].get_x()},
            {noeuds[permut[1]].get_y() - noeuds[permut[0]].get_y(),
              noeuds[permut[2]].get_y() - noeuds[permut[0]].get_y()}};
}

vector<vector<double>> Triangle::inv_mat_BT(void) {
    vector<vector<double>> BT = calc_mat_BT();
    return {{BT[1][1] / det, -BT[0][1] / det}, {-BT[1][0] / det, BT[0][0] / det}};
}

vector<vector<double>> Triangle::diff_terme(void) {
    vector<vector<double>> BI = inv_mat_BT();
    double d = det / 2;
    double d1 = BI[0][0] + BI[1][0];
    double d2 = BI[0][1] + BI[1][1];
    double m00 = (d1 * d1 + d2 * d2) * d;
    double m01 = (-BI[0][0] * d1 - BI[0][1] * d2) * d;
```

```

        double m02 = (-BI[1][0] * d1 - BI[1][1] * d2) * d;
        double m11 = (BI[0][0] * BI[0][0] + BI[0][1] * BI[0][1]) * d;
        double m12 = (BI[0][0] * BI[1][0] + BI[0][1] * BI[1][1]) * d;
        double m22 = (BI[1][0] * BI[1][0] + BI[1][1] * BI[1][1]) * d;
        return {{m00, m01, m02}, {m01, m11, m12}, {m02, m12, m22}};
    }

vector<vector<double>> Triangle::convect_terme(void) {
    double c = det / 6;
    return {{-c, -c, -c}, {c, c, c}, {0, 0, 0}};
}

vector<vector<double>> Triangle::react_terme(void) {
    double r1 = det / 12;
    double r2 = det / 24;
    return {{r1, r2, r2}, {r2, r1, r2}, {r2, r2, r1}};
}

```

## 5 Classe Maillage

### 5.1 header

```

#ifndef MAILLAGE_H
#define MAILLAGE_H

#include <vector>

#include "donnees_du_probleme.h"
#include "noeud.h"
#include "triangle.h"

using namespace std;

class Maillage {
    // Subdivisions.
    vector<Triangle> triangulation;

public:
    // Constructeur.
    Maillage(void);

    // Getters.
    vector<Triangle> get_triangulation(void);

    // Question 9:

```

```

// Retourne une subdivision uniforme de [-a, a] en N + 1 points et de [-b, b]
// en M + 1 points.
static vector<double> sub_div(double largeur, int nb_divisions);

// Question 11:
// Retourne le numéro global associé aux indices i et j.
int num_gb(int i, int j);

// Question 13:
// Retourne les indices i et j à partir du numéro global s.
vector<int> inv_num_gb(int s);

// Question 14:
// Retourne le numéro intérieur associé aux indices i et j.
int num_int(int i, int j);

// Question 16:
// Retourne les indices i et j à partir du numéro intérieur k.
vector<int> inv_num_int(int k);

// Question 17:
// Retourne le numéro global à partir du numéro intérieur k.
int num_int_gb(int k);

// Question 18:
// Retourne le numéro intérieur à partir du numéro global s.
int num_gb_int(int s);

// Retourne le numéro global d'un noeud dans le maillage.
int num_gb_noeud(Noeud noeud);

// Retourne le numéro intérieur d'un noeud dans le maillage.
int num_int_noeud(Noeud noeud);

// Vérifie si le noeud est sur le bord.
bool est_sur_le_bord(Noeud noeud);

// Donne les coordonnées x et y à partir du numéro intérieur du noeud.
vector<double> int_coord(int k);

// Question 20:
// Initialise le tableau de triangle dont la l-ème ligne contient le triangle
// T_l.
void init_maillage_TR(void);
};


```

#endif

## 5.2 implémentation

```
#include <cmath>

#include "maillage.h"

Maillage::Maillage(void) { this->init_maillage_TR(); }

vector<Triangle> Maillage::get_triangulation(void) { return triangulation; }

vector<double> Maillage::sub_div(double largeur, int nb_divisions) {
    vector<double> xi;
    for (int i = 0; i <= nb_divisions; i++)
        xi.push_back(-largeur + (2 * i * largeur) / nb_divisions);
    return xi;
}

int Maillage::num_gb(int i, int j) { return (N + 1) * j + i; }

vector<int> Maillage::inv_num_gb(int s) {
    int i = s % (N + 1);
    int j = s / (N + 1);
    return {i, j};
}

int Maillage::num_int(int i, int j) { return (N - 1) * (j - 1) + (i - 1); }

vector<int> Maillage::inv_num_int(int k) {
    int i = k % (N - 1) + 1;
    int j = k / (N - 1) + 1;
    return {i, j};
}

int Maillage::num_int_gb(int k) {
    vector<int> ij = this->inv_num_int(k);
    return this->num_gb(ij[0], ij[1]);
}

int Maillage::num_gb_int(int s) {
    vector<int> ij = this->inv_num_gb(s);
    return this->num_int(ij[0], ij[1]);
}

bool Maillage::est_sur_le_bord(Noeud noeud) {
```

```

        int i = round(N * (noeud.get_x() + a) / (2 * a));
        int j = round(M * (noeud.get_y() + b) / (2 * b));
        return i == 0 || j == 0 || i == N || j == M;
    }

int Maillage::num_gb_noeud(Noeud noeud) {
    int i = round(N * (noeud.get_x() + a) / (2 * a));
    int j = round(M * (noeud.get_y() + b) / (2 * b));
    return this->num_gb(i, j);
}

int Maillage::num_int_noeud(Noeud noeud) {
    int i = round(N * (noeud.get_x() + a) / (2 * a));
    int j = round(M * (noeud.get_y() + b) / (2 * b));
    return this->num_int(i, j);
}

vector<double> Maillage::int_coord(int k) {
    vector<int> ij = this->inv_num_int(k);
    return {(2 * ij[0] - N) * a / N, (2 * ij[1] - M) * b / M};
}

void Maillage::init_maillage_TR(void) {
    // On génère la matrice des noeuds.
    vector<vector<Noeud>> noeuds;
    for (int i = 0; i <= N; i++) {
        vector<Noeud> colonne;
        double x = (2 * i - N) * a / N;
        for (int j = 0; j <= M; j++) {
            double y = (2 * j - M) * b / M;
            colonne.push_back(Noeud(x, y));
        }
        noeuds.push_back(colonne);
    }
    for (int j = 0; j < M; j++) {
        for (int i = 0; i < N; i++) {
            Noeud S0 = noeuds[i][j];
            Noeud SE = noeuds[i + 1][j];
            Noeud NO = noeuds[i][j + 1];
            Noeud NE = noeuds[i + 1][j + 1];
            // Il y a 2 configurations possibles en fonction de la position du
            // rectangle du maillage qui nous intéresse. Dans chaque cas, il
            // faut déterminer les sommets du rectangle et les placer dans un
            // certain ordre.
            if ((i ^ j) & 1) {
                triangulation.push_back(Triangle(S0, SE, NO));
            }
        }
    }
}

```

```
        triangulation.push_back(Triangle(SE, NO, NE));
    } else {
        triangulation.push_back(Triangle(SO, NO, NE));
        triangulation.push_back(Triangle(SO, SE, NE));
    }
}
}
```

## 6 Boite à outils

## 6.1 header

```

#ifndef BOITE_A_OUTILS
#define BOITE_A_OUTILS

#include <vector>

#include "donnees_du_probleme.h"
#include "maillage.h"
#include "triangle.h"

using namespace std;

// On définit l'addition de 2 vecteur de double (ils doivent être de même
// taille).
vector<double> operator+(vector<double> A, vector<double> B);

// On définit la soustraction de 2 vecteur de double (ils doivent être de même
// taille).
vector<double> operator-(vector<double> A, vector<double> B);

// On définit le produit d'un vecteur de double avec un double.
vector<double> operator*(double scalaire, vector<double> B);

// On définit le produit scalaire de 2 vecteur de double (ils doivent être de
// même taille).
double operator*(vector<double> A, vector<double> B);

// Retourne la plus grande valeur absolue du vecteur.
double max(vector<double> A);

// Question 3:
// Retourne f_eta (x, y).
double f_second_membre(double (*u_g)(double), double (*u_d)(double),

```

```

        double (*u_gpp)(double), double (*u_dpp)(double),
        double x, double y);

// Question 43.d:
// Calcule la prolongation du vecteur des noeuds intérieurs sur tous les
// noeuds.
vector<double> extend_vec(vector<double> V_int);

// Question 43.e:
// Calcule la restriction du vecteur des noeuds globaux sur tous les noeuds
// intérieurs.
vector<double> int_vec(vector<double> V_glb);

// Question 46.i:
// Retourne la norme L2 de la fonction v associée au vecteur V de taille I.
double norme_L2(vector<double> V, Maillage maille);

// Question 46.j:
// Retourne la norme L2 grad de la fonction v associée au vecteur V de taille I.
double norme_L2_grad(vector<double> V, Maillage maille);

// Question 44:
// Retourne le produit vectoriel A_eta * V.
vector<double> mat_vec(vector<double> V, Maillage maille);

// Question 45:
// Retourne le second membre B_eta du système linéaire A_eta * X = B_eta.
vector<double> scd_membre(double (*rhfs)(double, double), Maillage maille);

// Partie 4:
// Retourne une solution approchée du système linéaire A_eta * X = B_eta.
vector<double> inv_syst(vector<double> B_eta, Maillage maille,
                        int max_iteration);

// Question 49:
// Retourne les trois erreurs relatives.
vector<double> erreurs(double (*sol_exa)(double, double),
                       vector<double> sol_appr, Maillage maille);

// TEMPORAIRE
double u_eta(double x, double y);

#endif

```

## 6.2 implémentation

```
#define _USE_MATH_DEFINES
#include <cmath>
#include <iostream>

#include "boite_a_outils.h"

using namespace std;

vector<double> operator+(vector<double> A, vector<double> B) {
    vector<double> C;
    for (size_t i = 0; i < B.size(); ++i)
        C.push_back(A[i] + B[i]);
    return C;
}

vector<double> operator-(vector<double> A, vector<double> B) {
    vector<double> C;
    for (size_t i = 0; i < B.size(); ++i)
        C.push_back(A[i] - B[i]);
    return C;
}

vector<double> operator*(double scalaire, vector<double> B) {
    vector<double> C;
    for (size_t i = 0; i < B.size(); ++i)
        C.push_back(scalaire * B[i]);
    return C;
}

double operator*(vector<double> A, vector<double> B) {
    double C;
    for (size_t i = 0; i < B.size(); ++i)
        C += A[i] * B[i];
    return C;
}

double max(vector<double> A) {
    double max = 0;
    for (double i : A)
        max = i > 0 ? i > max ? i : max : -i > max ? -i : max;
    return max;
}

double f_second_membre(double (*u_g)(double), double (*u_d)(double),
```

```

        double (*u_gpp)(double), double (*u_dpp)(double),
        double x, double y) {
    return (epsilon * (a - x) * u_gpp(y) + epsilon * (a + x) * u_dpp(y) +
            gama * u_g(y) - gama * u_d(y) - lambda * (a - x) * u_g(y) -
            lambda * (a + x) * u_d(y)) /
        (2 * a);
}

vector<double> extend_vec(vector<double> V_int) {
    vector<double> V_glb;
    for (int i = 0; i <= N; i++)
        V_glb.push_back(0);
    int numInt = 0;
    for (int j = 1; j < M; j++) {
        V_glb.push_back(0);
        for (int i = 1; i < N; i++) {
            V_glb.push_back(V_int[numInt]);
            numInt++;
        }
        V_glb.push_back(0);
    }
    for (int i = 0; i <= N; i++)
        V_glb.push_back(0);
    return V_glb;
}

vector<double> int_vec(vector<double> V_glb) {
    vector<double> V_int;
    for (int j = 1; j < M; j++) {
        for (int i = 1; i < N; i++)
            V_int.push_back(V_glb[(N + 1) * j + i]);
    }
    return V_int;
}

double norme_L2(vector<double> V, Maillage maille) {
    vector<double> V_glb = extend_vec(V);
    vector<double> WW((N + 1) * (M + 1), 0);
    for (Triangle triangle : maille.get_triangulation()) {
        vector<Noeud> noeuds = triangle.get_noeuds();
        for (int i = 0; i < 3; i++) {
            int s = maille.num_gb_noeud(noeuds[i]);
            double res = 0;
            for (int j = 0; j < 3; j++) {
                int r = maille.num_gb_noeud(noeuds[j]);
                res += V_glb[r] * triangle.react_terme()[j][i];
            }
        }
    }
}
```

```

        }
        WW[s] += res;
    }
}
vector<double> AV = int_vec(WW);
return AV * V;
}

double norme_L2_grad(vector<double> V, Maillage maille) {
    vector<double> V_glb = extend_vec(V);
    vector<double> WW((N + 1) * (M + 1), 0);
    for (Triangle triangle : maille.get_triangulation()) {
        vector<Noeud> noeuds = triangle.get_noeuds();
        for (int i = 0; i < 3; i++) {
            int s = maille.num_gb_noeud(noeuds[i]);
            double res = 0;
            for (int j = 0; j < 3; j++) {
                int r = maille.num_gb_noeud(noeuds[j]);
                res += V_glb[r] * triangle.diff_terme()[j][i];
            }
            WW[s] += res;
        }
    }
    vector<double> AV = int_vec(WW);
    return AV * V;
}

vector<double> mat_vec(vector<double> V, Maillage maille) {
    vector<double> VV = extend_vec(V);
    vector<double> WW((N + 1) * (M + 1), 0);
    for (Triangle triangle : maille.get_triangulation()) {
        vector<Noeud> noeuds = triangle.get_noeuds();
        for (int i = 0; i < 3; i++) {
            int s = maille.num_gb_noeud(noeuds[i]);
            double res = 0;
            for (int j = 0; j < 3; j++) {
                int r = maille.num_gb_noeud(noeuds[j]);
                double prod2 = epsilon * triangle.diff_terme()[j][i] +
                    gama * triangle.convect_terme()[j][i] +
                    lambda * triangle.react_terme()[j][i];
                res += VV[r] * prod2;
            }
            WW[s] += res;
        }
    }
    return int_vec(WW);
}

```

```

}

vector<double> scd_membre(double (*rhfs)(double, double), Maillage maille) {
    vector<double> B((N - 1) * (M - 1), 0);
    for (Triangle triangle : maille.get_triangulation()) {
        vector<Noeud> noeuds = triangle.get_noeuds();
        for (int i = 0; i < 3; i++) {
            if (!maille.est_sur_le_bord(noeuds[i])) {
                vector<vector<double>> BT =
                    triangle.calc_mat_BT({i, (i + 1) % 3, (i + 2) % 3});
                double res = 0;
                //  $FT(1/2, 0) = BT * (1/2, 0) + (x_0, y_0)$ :
                vector<double> FT = {BT[0][0] / 2 + noeuds[i].get_x(),
                                      BT[1][0] / 2 + noeuds[i].get_y()};
                //  $wk(FT(1/2, 0)) = 1/2$ 
                res += rhfs(FT[0], FT[1]) / 12;
                //  $FT(0, 1/2) = BT * (0, 1/2) + (x_0, y_0)$ :
                FT = {BT[0][1] / 2 + noeuds[i].get_x(),
                      BT[1][1] / 2 + noeuds[i].get_y()};
                //  $wk(FT(0, 1/2)) = 1/2$ 
                res += rhfs(FT[0], FT[1]) / 12;
                B[maille.num_int_noeud(noeuds[i])] += res;
            }
        }
        return B;
    }
}

vector<double> inv_syst(vector<double> B_eta, Maillage maille,
                        int max_iteration) {
    vector<double> X0(B_eta.size(), 1);
    vector<double> R0 = B_eta - mat_vec(X0, maille);
    vector<double> R0_etoile = R0;
    vector<double> W0 = R0;
    for (int j = 0; j < max_iteration; j++) {
        vector<double> AW0 = mat_vec(W0, maille);
        double alpha0 = (R0 * R0_etoile) / (AW0 * R0_etoile);
        vector<double> S0 = R0 - (alpha0 * AW0);
        vector<double> AS0 = mat_vec(S0, maille);
        double omega0 = (AS0 * S0) / (AS0 * AS0);
        vector<double> X1 = X0 + (alpha0 * W0) + (omega0 * S0);
        vector<double> R1 = S0 - (omega0 * AS0);
        double beta0 = ((R1 * R0_etoile) / (R0 * R0_etoile)) * (alpha0 / omega0);
        vector<double> W1 = R1 + (beta0 * (W0 - (omega0 * AW0)));
        R0 = R1;
        W0 = W1;
    }
}

```

```

        X0 = X1;
    }
    return X0;
}

vector<double> erreurs(double (*sol_exa)(double, double),
                      vector<double> sol_appr, Maillage maille) {
    vector<double> w;
    int I = (N - 1) * (M - 1);
    for (int k = 0; k < I; k++) {
        vector<double> xy = maille.int_coord(k);
        w.push_back(sol_exa(xy[0], xy[1]));
    }
    vector<double> erreur = w - sol_appr;
    return {norme_L2(erreur, maille) / norme_L2(w, maille),
            norme_L2_grad(erreur, maille) / norme_L2_grad(w, maille),
            max(erreur) / max(w)};
}

double u_eta(double x, double y) {
    double A = (gama / epsilon - sqrt(gama * gama / epsilon / epsilon +
                                         4 * M_PI * M_PI + 4 * lambda / epsilon)) /
               2;
    double B = (gama / epsilon + sqrt(gama * gama / epsilon / epsilon +
                                         4 * M_PI * M_PI + 4 * lambda / epsilon)) /
               2;
    double C_1 = 1 / (exp(-A) - exp(A - 2 * B));
    double C_2 = 1 / (exp(-B) - exp(B - 2 * A));
    double U_x = C_1 * exp(A * x) + C_2 * exp(B * x);
    return U_x * sin(M_PI * y);
}

```