Solution TP3: Approximation des EDP de type elliptique avec Matlab

Préparée par : Pr L. Alem et Dr I. Djerrar

TP: Un problème raide

On considère le problème de type elliptique suivant:

$$\begin{cases} -\varepsilon u_{xx}(x) + a u_x(x) + b u(x) = f(x), \quad x \in (0,1) \\ u(0) = c, \\ u(1) = d. \end{cases} \tag{1}$$

Où $\varepsilon,\ a,\ b,\ c,\ d,$ sont des réels avec $\varepsilon > 0$ et f une fonction continue sur $[0,\ 1]$. On admettra que le problème (1) admet une solution unique $u \in C^2([0,1];\mathbb{R})$. On se propose de discrétiser le problème (1) par la méthode de différences finies. Pour cela on prend une grille uniforme $0 = x_0 < x_1 < \cdots < x_n = 1$ de pas $h = \dfrac{1}{n}, \quad n \geq 1.$

- Le schéma de discrétisation du problème (1)

On note par $u(x_i) = u_i$ et $f(x_i) = f_i, \quad i = 1, \cdots, n-1$

1. En approchant la dérivée seconde u_{xx} par un schéma à 3 points

$$u_i'' \simeq \frac{u_{i-1} - 2u_i + u_{i+1}}{h^2}, \quad i = 1 \cdots n-1$$

2. En approchant la dérivée seconde u_x par un schéma centré

$$u_i' \simeq \frac{u_{i+1} - u_{i-1}}{2h}, \quad i = 1 \cdots n-1$$

3. Les conditions aux limites

$$\begin{cases} u(0) = u(x_0) = u_0 = c \\ u(1) = u(x_n) = u_n = d \end{cases}$$

Le problème discret s'écrit:

$$\begin{cases} -(\dfrac{a}{2h} + \dfrac{\varepsilon}{h^2})u_{i-1} + (\dfrac{2\varepsilon}{h^2} + b)u_i + (\dfrac{a}{2h} - \dfrac{\varepsilon}{h^2})u_{i+1} = f_i, \quad i = 1 \cdots n-1 \\ u_0 = c, \\ u_n = d. \end{cases} \tag{2}$$

- L'écriture matricielle sous la forme $AU = b$

$$\begin{cases} i = 1, \quad -(\dfrac{a}{2h} + \dfrac{\varepsilon}{h^2})u_0 + (\dfrac{2\varepsilon}{h^2} + b)u_1 + (\dfrac{a}{2h} - \dfrac{\varepsilon}{h^2})u_2 = f_1 \\ i = 2, \quad -(\dfrac{a}{2h} + \dfrac{\varepsilon}{h^2})u_1 + (\dfrac{2\varepsilon}{h^2} + b)u_2 + (\dfrac{a}{2h} - \dfrac{\varepsilon}{h^2})u_3 = f_2 \\ \vdots \\ i = n-1, \quad -(\dfrac{a}{2h} + \dfrac{\varepsilon}{h^2})u_{n-2} + (\dfrac{2\varepsilon}{h^2} + b)u_{n-1} + (\dfrac{a}{2h} - \dfrac{\varepsilon}{h^2})u_n = f_{n-1} \end{cases} \tag{3}$$

On remplace $u_0 = c$ et $u_n = d$ dans (3), on obtient

$$\begin{cases} i = 1, & (\frac{2\varepsilon}{h^2} + b)u_1 + (\frac{a}{2h} - \frac{\varepsilon}{h^2})u_2 = f_1 + (\frac{\varepsilon}{h^2} + \frac{a}{2h})c \\ i = 2, & -(\frac{a}{2h} + \frac{\varepsilon}{h^2})u_1 + (\frac{2\varepsilon}{h^2} + b)u_2 + (\frac{a}{2h} - \frac{\varepsilon}{h^2})u_3 = f_2 \\ \vdots & \\ i = n-1, & -(\frac{a}{2h} + \frac{\varepsilon}{h^2})u_{n-2} + (\frac{2\varepsilon}{h^2} + b)u_{n-1} = f_{n-1} - (\frac{a}{2h} - \frac{\varepsilon}{h^2})d \end{cases}$$

d'où le système linéaire $AU = b$, avec $U = (u_1, \cdots, u_{n-1})^T$ désigne le vecteur constitué de la solution approchée aux noeuds x_i.
b est le vecteur second membre, tel que

$$b = \left(f_1 + (\frac{\varepsilon}{h^2} + \frac{a}{2h})c, \quad f_2, \cdots, f_{n-2}, \quad f_{n-1} - (\frac{a}{2h} - \frac{\varepsilon}{h^2})d\right)^T.$$

Et la matrice A (matrice tridiagonale) de taille $(n-1)^2$

$$A = \begin{pmatrix} \alpha & \beta & & & \\ \gamma & \alpha & \beta & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \beta \\ & & & \gamma & \alpha \end{pmatrix}$$

avec

$$\begin{cases} \alpha = \dfrac{2\varepsilon}{h^2} + b \\ \beta = \dfrac{a}{2h} - \dfrac{\varepsilon}{h^2} \\ \gamma = -(\dfrac{a}{2h} + \dfrac{\varepsilon}{h^2}) \end{cases}$$

- Le système admet une solution unique :
La matrice A n'est pas symétrique, donc on ne peux pas appliquer le théorème (symétrique+définie positive $\implies$ inversible).
Il faut montrer que A est monotone i.e

$$Au \geq 0 \implies u \geq 0$$

En déduire que A est inversible. (Voir le cours).

- Le script Matlab raide.m :

```
function [x,uh]=raide(e,a,b,c,d,n,f)
% e = epsilon
clc;clf;
h=1/n;
x=0:h:1;
w = ((-e/(h^2)) - a/(2*h)) * ones(n-2,1);  % sous la diagonale
w1 = ((2*e/(h^2)) + b) * ones(n-1,1);  % la diagonale
w2 = ((a/(2*h)) - e/(h^2)) * ones(n-2,1)  % sur la diagonale
Ah = diag(w1) + diag(w,-1) + diag(w2,1);
```

$disp(Ah)$,
$bh = [(e/(h^2) + a/(2*h)) * c + f(1); f(2:n-2); -(a/(2*h) + e/(h^2)) * d + f(n-1)]$; % si f est une constante
% $bh = [(e/(h^2) + a/(2*h)) * c + f(x(1)); f(x(2:n-2)); -(a/(2*h) + e/(h^2)) * d + f(x(n-1))]$; % si f est une fonction en x
bh,
$uh = Ah \setminus bh$;
uh=[c; uh; d]; % on ajoute les conditions aux limites u_0 et u_n
plot(x,uh);

Application:

Exemple 1:

- On prend $\varepsilon = 1, a = b = c = d = 0$ et $f \equiv 1$,

On aura

$$\begin{cases} -u_{xx}(x) = 1, \quad x \in (0,1) \\ u(0) = 0, \\ u(1) = 0. \end{cases}$$

La solution exacte est donnée par $u(x) = \dfrac{x - x^2}{2}$

Exemple 2:

- On prend $\varepsilon = \dfrac{1}{4}, a = 1,\ b = 3,\ c = 1,\ d = e^{-2}$ et $f \equiv 0$

On aura

$$\begin{cases} -\dfrac{1}{4}u_{xx}(x) + u_x(x) + 3u(x) = 0, \quad x \in (0,1) \\ u(0) = 1, \\ u(1) = e^{-2}. \end{cases}$$

La solution exacte est donnée par $u(x) = e^{-2x}$

Exemple 3:

- On prend $\varepsilon = \dfrac{1}{100}, a = 1,\ b = 0,\ c = 0,\ d = 1$ et $f \equiv 0$

On aura

$$\begin{cases} -\dfrac{1}{100}u_{xx}(x) + u_x(x) = 0, \quad x \in (0,1) \\ u(0) = 0, \\ u(1) = 1. \end{cases}$$

La solution exacte est donnée par $u(x) = \dfrac{e^{100x} - 1}{e^{100} - 1}$

- Pour tracer la solution approchée et la solution exacte pour chaque exemple, on utilise le script raide.m, on ajoutant les commandes nécessaires.

Le script:

```
function [x,uh]=raide(e,a,b,c,d,n,f)
clc;clf;
% e = epsilon
h=1/n;
x=0:h:1;
w = ((-e/(h^2)) - a/(2*h)) * ones(n-2,1);  % sous la diagonale
w1 = ((2*e/(h^2)) + b) * ones(n-1,1);  % la diagonale
w2 = ((a/(2*h)) - e/(h^2)) * ones(n-2,1)  % sur la diagonale
ah = diag(w1) + diag(w,-1) + diag(w2,1);
disp(ah),
bh = [(e/(h^2) + a/(2*h))*c + f(1); f(2:n-2); -(a/(2*h) + e/(h^2))*d + f(n-1)];
bh,
uh = ah \ bh;
uex = @(x)(1/2)*(x - x^2);% sol exacte pour le 1er exemple
%uex = @(x)exp(-2*x); % sol exacte pour le 2ème exemple
%uex = @(x)(exp(100*x)-1)/(exp(100)-1); % sol exacte pour 3ème exemple
uh=[c; uh; d];
plot(x,uh);
hold on
fplot(uex,[0,1],'r')
legend( 'sol approch', 'sol exacte' )
```

Exécution :

Exemple 1:

```
>> e = 1;
>> a = 0;
>> b = 0;
>> c = 0;
>> d = 0;
>> n = 10;
>> f = ones(n-1,1);
>> [x,uh] = raide(e,a,b,c,d,n,f)
```

Voir figure 1

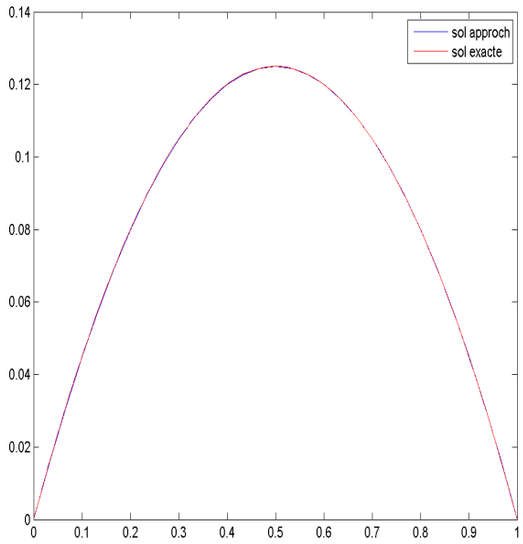


Figura 1: Exemple 1: La solution exacte et la solution approchée.

Exemple 2:

$>> e = 1/4;$
$>> a = 1;$
$>> b = 3;$
$>> c = 1;$
$>> d = exp(-2);$
$>> n = 10;$
$>> f = zeros(n-1, 1);$
$>> [x, uh] = raide(e, a, b, c, d, n, f)$
Voir figure 2

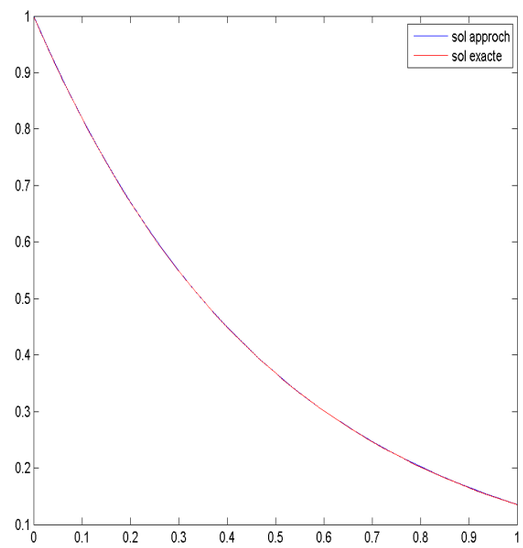
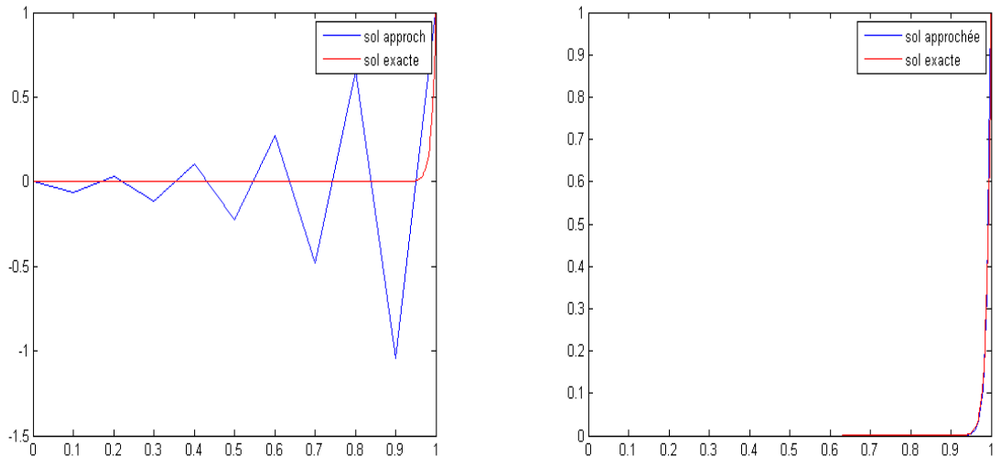


Figura 2: Exemple 2: La solution exacte et la solution approchée

Exemple 3:

$>> e = 1/100;$
$>> a = 1;$

$>> b = 0;$
$>> c = 0;$
$>> d = 1;$
$>> n = 10;$
$>> f = zeros(n-1, 1);$
$>> [x, uh] = raide(e, a, b, c, d, n, f)$
Voir figure 3



a) b)

Figura 3: Exemple 3: La solution exacte et la solution approchée, a) avec n=10, b) avec n=100.

Remarque:
On remarque que lorsque ε est très petit, la solution exacte est difficile à approcher.

- Ordre de consistance

Le schéma de discrétisation (2) est dit consistant d'ordre $p \geq 1$ si

$$\max_{1 \leq i \leq n-1} |R_i| = o(h^p)$$

R_i: erreur de consistance.
Le script err_raide.m
function err=err_raide(e,a,b,c,d,uex)
clc;clf;
for i=70:30:700
$[x, uh] = raide(e, a, b, c, d, i, ones(i-1, 1));$
%[x,uh]=raide(e,a,b,c,d,i,zeros(i-1,1)); % si f=0
u=uex(x)';
err(i)=norm(u-uh,inf);
end
plot(log(70:30:700),log(err(70:30:700)))

- Le script err_raide.m fait appel au script raide.m (les deux scripts soient dans le même répertoire)

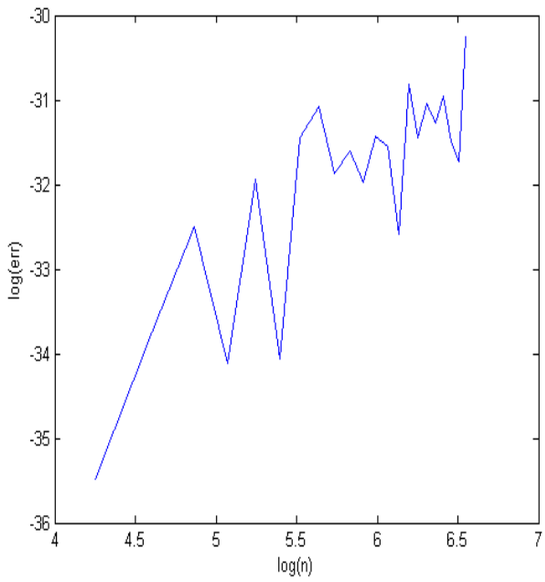


Figura 4: L'erreur en norme $\| \|_\infty$ pour le 1er exemple

Exécution:

Exemple 1:
$>> e = 1;$
$>> a = 0;$
$>> b = 0;$
$>> c = 0;$
$>> d = 0;$
$>> uex = @(x)(1/2) * (x - x^2);$
$>> err = err\_raide(e, a, b, c, d, uex)$
Voir figure 4.

Exemple 2:

$>> e = 1/4;$
$>> a = 1;$
$>> b = 3;$
$>> c = 1;$
$>> d = exp(-2);$
$>> uex = exp(-2 * x)$
$>> err = err\_raide(e, a, b, c, d, uex)$
Voir figure 5.

Exemple 3:

$>> e = 1/100;$
$>> a = 1;$
$>> b = 0;$
$>> c = 0;$
$>> d = 1;$
$>> uex = @(x)(exp(100 * x) - 1)./(exp(100) - 1);$

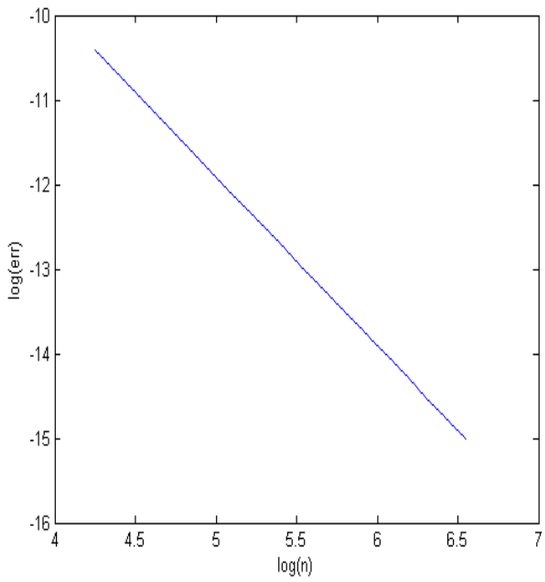


Figura 5: L'erreur en norme $\| \|_\infty$ pour le 2ème exemple

$>> err = err\_raide(e, a, b, c, d, uex)$
Voir figure 6.

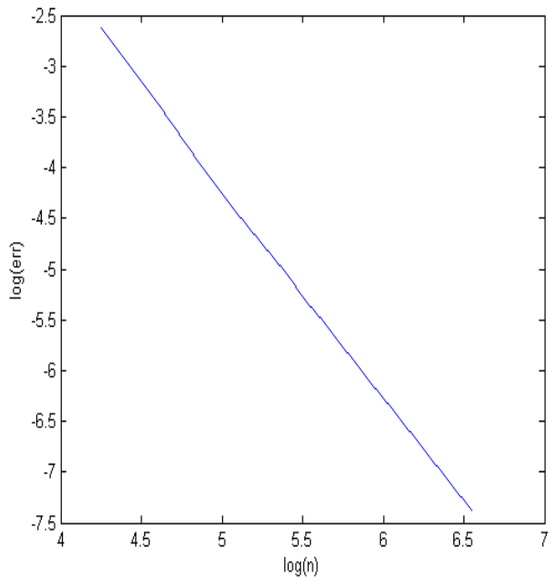


Figura 6: L'erreur en norme $\| \ \|_\infty$ pour le 3ème exemple

Remarque:
En calculant la pente p pour les exemples 2 et 3 avec la commande **polyfit** comme suit:
$p = polyfit(log(70 : 30 : 700), log(err(70 : 30 : 700)), 1)$ % on ajoute cette commande dans le script err_raide.m
On trouve:
$p = -1,9999$ pour l'exemple 2.
$p = -2{,}0495$ pour l'exemple 3.
Alors le schéma de discrétisation est consistant d'ordre 2.