# Corrigé 16

## Exercice 1

**1.a)** On pose $u_i^0 = \sin(\pi x_i),\ i = 0, ..., N+1$. Pour tout $n \geqslant 0$, étant donné $u_i^n,\ i = 0, ..., N+1$, le problème discrétisé en temps et en espace par la méthode d'Euler rétrograde revient à chercher les $u_i^{n+1},\ i = 0, ..., N+1$ tels que

$$\left\{ \begin{array}{rcll} \dfrac{u_i^{n+1} - u_i^n}{\tau} - \epsilon \dfrac{u_{i-1}^{n+1} - 2u_i^{n+1} + u_{i+1}^{n+1}}{h^2} + c_0 \dfrac{u_i^{n+1} - u_{i-1}^{n+1}}{h} & = & f(x_i, t_{n+1}), & i = 1, ..., N, \\ u_0^{n+1} = u_{N+1}^{n+1} & = & 0. & \end{array} \right.$$

La relation ci-dessus nous permet également d'écrire :

$$u_i^{n+1}\left(1 + \frac{2\epsilon\tau}{h^2} + \frac{c_0\tau}{h}\right) + u_{i-1}^{n+1}\left(-\frac{\epsilon\tau}{h^2} - \frac{c_0\tau}{h}\right) - \frac{\epsilon\tau}{h^2}u_{i+1}^{n+1} = u_i^n + \tau f(x_i, t_{n+1}), \quad i = 1, \ldots, N. \tag{1}$$

On a le schéma sous la forme $A\vec{u}^{n+1} = \vec{u}^n + \tau\vec{f}(t_{n+1})$ avec :

$$A = \begin{pmatrix} 1 + \dfrac{2\tau\epsilon}{h^2} + \dfrac{\tau c_0}{h} & -\dfrac{\tau\epsilon}{h^2} & & \\ -\dfrac{\tau\epsilon}{h^2} - \dfrac{\tau c_0}{h} & \ddots & \ddots & \\ & \ddots & \ddots & -\dfrac{\tau\epsilon}{h^2} \\ & & -\dfrac{\tau\epsilon}{h^2} - \dfrac{\tau c_0}{h} & 1 + \dfrac{2\tau\epsilon}{h^2} + \dfrac{\tau c_0}{h} \end{pmatrix},$$

**1.b)** Le fichier MATLAB `convdiff.m` est complété de la manière suivante :

```
function [u]=convdiff(N,M,tau)
h=1/(N+1);
t=0;
epsilon = 1;
c0 = 10;
%
% condition initiale
%
for i=1:N
  u(i)= sin (pi*i*h);
end
%
% remplissage de la matrice  A
%
for i=1:N
  a(i) =1+(2*tau*epsilon)/(h*h)+(tau*c0)/h;
end
for i=1:N-1
  d(i) =  (-tau*epsilon)/(h*h) -(tau*c0)/h ;
end
for i=1:N-1
  c(i) =  (-tau*epsilon)/(h*h);
end
%
% decomposition LU  A = LU
%
c(1)=c(1)/a(1);
for i=2:N-1
  a(i) = a(i)-c(i-1)*d(i-1);
  c(i) = c(i)/a(i);
end
 a(N) = a(N)-c(N-1)*d(N-1);
%
% schema d'Euler retrograde : A u^n+1 = u^n + tau f^{n+1}
```

```
%
for n=1:M
  t=t+tau;
%
% second membre du systeme lineaire u^n + tau f^{n+1}
%
  for i=1:N
     u(i) =  u(i)+tau*10*pi*cos(pi*i*h)*exp(-pi*pi*t);
  end
%
% resolution du systeme lineaire Ly = u^n + tau f^{n+1}
%
  u(1)=u(1)/a(1);
  for i=1:N-1
    u(i+1) = (u(i+1)-d(i)*u(i))/a(i+1);
  end
%
% resolution du systeme lineaire U u^n+1 = y
%
  for i=N-1:-1:1
    u(i) = (u(i)-c(i)*u(i+1));
  end
end
%Calcul de l'erreur
for i=1:N-1
   err(i)=abs(u(i)-uex(i*h,M*tau));
end
supererror=max(err)
function u_exact = uex(x,t)
u_exact = exp(-pi*pi*t)*sin(pi*x);
```

**1.c)** On a les résultats suivants au temps $t = 0.2$ :

| $N+1$ | h | $\tau$ | M | Erreur |
|---|---|---|---|---|
| 10 | 0.1 | 0.02 | 10 | 3.614635e-02 |
| 20 | 0.05 | 0.01 | 20 | 2.012016e-02 |
| 40 | 0.025 | 0.005 | 40 | 1.059199e-02 |
| 80 | 0.0125 | 0.0025 | 80 | 5.436055e-03 |

On note que, pour $h$ et $\tau$ suffisamment petits, l'erreur est approximativement divisée par deux lorsque $h$ est divisé par deux et $\tau$ est divisé par deux. Le schéma est donc bien d'ordre $h + \tau$.

**1.d)** Pour $n$ fixé, notons k un entier tel que

$$u_k^{n+1} \leqslant u_j^{n+1} \qquad j = 0, 1, 2, ..., N+1.$$

Supposons pour commencer que $k$ est différent de 0 et que $k$ est différent de $N+1$. En prenant $i = k$ et $f(x,t) = 0$ dans (1), nous obtenons :

$$\left(1 + \frac{2\epsilon\tau}{h^2} + \frac{c_0\tau}{h}\right) u_k^{n+1} = u_{k-1}^{n+1}\left(\frac{\epsilon\tau}{h^2} + \frac{c_0\tau}{h}\right) + \frac{\epsilon\tau}{h^2}u_{k+1}^{n+1} + u_k^n.$$

Puisque $u_{k-1}^{n+1} \geqslant u_k^{n+1}$ et $u_{k+1}^{n+1} \geqslant u_k^{n+1}$, nous avons donc :

$$\left(1 + \frac{2\epsilon\tau}{h^2} + \frac{c_0\tau}{h}\right) u_k^{n+1} \geqslant \left(\frac{\epsilon\tau}{h^2} + \frac{c_0\tau}{h}\right) u_k^{n+1} + \frac{\epsilon\tau}{h^2}u_k^{n+1} + u_k^n.$$

Et par suite $u_k^{n+1} \geqslant u_k^n \geqslant 0$.
Si $k = 0$ ou $k = N+1$, alors la relation ci-dessus est trivialement vraie. Si nous supposons que tous les $u_j^n$ sont positifs ou nuls, nous obtenons donc :

$$0 \leqslant u_k^n \leqslant u_k^{n+1} \leqslant u_j^{n+1} \qquad j = 0, 1, 2, ..., N+1,$$

ce qui prouve que tous les $u_j^{n+1}$ sont aussi positifs ou nuls.

# Exercice 2

**1.a)**
On commence par calculer $\frac{\partial F}{\partial a_k}(\vec{a}, \vec{\omega}, \vec{b})$ :

$$\frac{\partial F}{\partial a_k}(\vec{a},\vec{\omega},\vec{b}) = \frac{\partial}{\partial a_k}\left(\frac{1}{2}\sum_{i=1}^{m}\left(\frac{1}{n}\sum_{j=1}^{n} a_j \,\mathrm{relu}(\omega_j x_i + b_j) - f(x_i)\right)^2\right)$$

$$= \sum_{i=1}^{m}\left(\frac{1}{n}\sum_{j=1}^{n} a_j \,\mathrm{relu}(\omega_j x_i + b_j) - f(x_i)\right)\cdot \frac{1}{n}\mathrm{relu}(\omega_k x_i + b_k)$$

Soit $A \in \mathbb{R}^{m\times n}$, $\vec{f} \in \mathbb{R}^m$, pour $k = 1,\ldots,n$ on peut écrire :

$$\left(A^T(A\vec{a} - \vec{f})\right)_k = \sum_{i=1}^{m} A_{ik}\left(\sum_{j=1}^{n} A_{ij}a_j - f_i\right)$$

En définissant pour $i = 1,\ldots,m$, $k = 1,\ldots,n$ : $A_{ik} = \frac{1}{n}\mathrm{relu}(\omega_k x_i + b_k)$ et $f_i = f(x_i)$, on obtient le résultat désiré. De façon similaire, on obtient :

$$W_{ik} = \frac{1}{n}a_k \,\mathrm{relu}'(\omega_k x_i + b_k)x_i, \quad B_{ik} = \frac{1}{n}a_k \,\mathrm{relu}'(\omega_k x_i + b_k),$$

où $\mathrm{relu}'(x) = 1$ si $x > 0$, et 0 sinon.

**1.b)**

Le fichier est complété de la manière suivante :

```
function [a,w,b]=twolayersnorm(n,m,step,itmax)
pkg load statistics
% try with [a,w,b]=twolayersnorm(100,20,1,10000);
% goal: approach any function by sum_j a_j relu(w_j x + b_j)
% n: nb of relu functions
% m: nb of interpolation points
% a: contains the a_j
% w: contains the w_j
% b: contains the b_j

x=sparse(m,1);
y=sparse(m,1);
for i=1:m
    x(i)=-1+2*i/(m+1);
    y(i)=exp(-10*x(i)^2);
end

%a=(0,1,n,1);
%w=normrnd(0,1,n,1);
%b=normrnd(0,1,n,1);

a=-1+2*rand(n,1);
w=-1+2*rand(n,1);
b=-1+2*rand(n,1);
mata=sparse(m,n);
matw=sparse(m,n);
matb=sparse(m,n);

% gradient descent
for iter=1:itmax
    for i=1:m
        for j=1:n
            mata(i,j)=relu(w(j)*x(i)+b(j));
            matb(i,j)=a(j)*relup(w(j)*x(i)+b(j));
            matw(i,j)=matb(i,j)*x(i);
        end
    end
    mata=mata/n;
    matb=matb/n;
    matw=matw/n;
    res=mata*a-y;
    grada=mata'*res;
    gradw=matw'*res;
    gradb=matb'*res;
    a=a-step*grada;
    w=w-step*gradw;
    b=b-step*gradb;
    if (mod(iter,10)==0)
        fprintf(" iter %d norm(grada) %e norm(gradw) %e norm(gradb) %e \n",iter,norm(grada),norm(gradw),norm(gradb));
        plot(x,y,'+',x,mata*a,'x',-b./w,zeros(n,1),'o');
        xlim([-5 5]);
        ylim([-0.5 1]);
        refresh();
```

```
    endif
end
end

function relu=relu(x)
    if (x<0)
        relu=0.;
    else
        relu=x;
    endif
end

function relup=relup(x)
    if (x<=0)
        relup=0.;
    else
        relup=1;
    endif
end
```

Dans la figure suivante, on observe l'approximation de la fonction. La convergence est très lente.