ÉCOLE CENTRALELYON

Année 2018-2019

Analyse numérique – TD 1

Analyse numérique matricielle

À préparer avant la séance de TD

Exercice 1. Soit $A \in \mathbb{R}^{d\times d}$ une matrice symétrique définie positive ; on note ses valeurs propres $0 < \lambda_1 \leqslant \lambda_2 \leqslant \cdots \leqslant \lambda_d$. On considère la méthode itérative, pour $\alpha > 0$,

$$X^{(n+1)} = X^{(n)} - \alpha(AX^{(n)} - b), \qquad X^{(0)} \in \mathbb{R}^d \text{ donné.} \tag{1}$$

1. Mettre la méthode sous la forme
$$X^{(n+1)} = BX^{(n)} + c,$$
où l'on précisera la matrice B et le vecteur c.
2. On suppose que $\alpha\lambda_d < 2$. Montrer que toutes les valeurs propres de B appartiennent à l'intervalle $]-1, 1[$.
3. En déduire que, sous la condition $\alpha\lambda_d < 2$, la suite $\left(X^{(n)}\right)$ converge. Quelle est sa limite ?

*4. Pour quelle valeur de α la convergence est-elle la plus rapide ?

Correction.

1. *La méthode s'écrit $X^{(n+1)} = BX^{(n)} + c$, avec $B = I - \alpha A$ et $c = \alpha b$.*
2. *Les valeurs propres de B sont $\mu_i = 1 - \alpha\lambda_i$ pour $i = 1, 2, \ldots, d$. Comme A est définie positive, elles sont toutes strictement inférieures à 1. Par ailleurs, pour chaque i,*
$$1 - \alpha\lambda_i \geqslant 1 - \alpha\lambda_d \geqslant -1 \text{ car } \alpha\lambda_d < 2.$$
3. *Ainsi, on a $\rho(B) < 1$, donc la suite $X^{(n)}$ converge. Notons X sa limite ; en passant à la limite dans l'inégalité (1), on obtient $AX = b$.*
4. *Il s'agit de calculer le rayon spectral de la matrice d'itération. D'après la discussion ci-dessus,*
$$\rho(B) = \max_{i=1,2,\ldots d} |\mu_i| = \max(1 - \alpha\lambda_1, \alpha\lambda_d - 1)$$
Cette quantité est minimale pour $1 - \alpha\lambda_1 = \alpha\lambda_d - 1$ (faire un dessin !), qui fournit le paramètre α optimal :
$$\alpha = \frac{2}{\lambda_1 + \lambda_d}.$$

Exercice 2. Programmer la méthode (1) avec `matlab`. La tester pour la matrice

$$A = \begin{pmatrix} 2 & -1 & & \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & -1 \\ & & -1 & 2 \end{pmatrix}.$$

On prendra $b = (1, 1, \ldots, 1)^\intercal$, $X^{(0)} = (0, 0, \ldots, 0)^\intercal$ et les paramètres $d = 10$, $\alpha = 10^{-2}$. Vérifier que la valeur de la septième composante de $X^{(10000)}$ vaut 13.9957.

Correction. *Le code* `matlab` *est le suivant*

```
% Parametres
d = 10;
alpha = 1e-2;

% Cas test
un = ones(d-1,1);
A = 2*eye(d) - diag(un,1) - diag(un,-1);
b = ones(d,1);
Xex = A\b;

% Algorithme (gradient a pas fixe)
X = zeros(d,1);
for i = 1:10000
        X = X - alpha*(A*X - b);
end

% Application numerique
X(7)
```

À travailler pendant la séance de TD

Exercice 3. On considère la matrice tridiagonale

$$A = \begin{pmatrix} a_1 & c_1 & & & \\ b_1 & \ddots & \ddots & & \\ & \ddots & & c_{d-1} & \\ & & b_{d-1} & a_d \end{pmatrix}$$

On suppose que A est inversible et qu'elle admet une décomposition LU sous la forme

$$L = \begin{pmatrix} 1 & & \\ \ell_1 & \ddots & \\ & \ell_{d-1} & 1 \end{pmatrix} \qquad U = \begin{pmatrix} u_1 & v_1 & & \\ & \ddots & \ddots & \\ & & & v_{d-1} \\ & & & u_d \end{pmatrix}.$$

1. Déterminer les relations permettant de construire (u_i),(v_i), (ℓ_i) en fonction de (a_i), (b_i), (c_i).

2. En déduire un algorithme de calcul de la décomposition LU de la matrice A. Quel est son coût ? Comment se compare-t-il au coût dans le cas général ?

Correction.

1. *On procède par identification ; le produit LU fournit*

$$LU = \begin{pmatrix} u_1 & v_1 & & & \\ \ell_1 u_1 & \ell_1 v_1 + u_2 & \ddots & & \\ & \ddots & \ddots & v_{d-1} & \\ & & \ell_{d-1} u_{d-1} & \ell_{d-1} v_{d-1} + u_d \end{pmatrix},$$

si bien qu'on en déduit les relations : $u_1 = a_1$ et, pour $i = 1, 2, \dots d-1$,

$$u_{i+1} = a_{i+1} - \frac{b_i}{u_i} v_i, \qquad \ell_i = \frac{b_i}{u_i}, \qquad v_i = c_i.$$

La décomposition LU existe si $u_i \neq 0$ pour tout i.

2. *Notons $a \in \mathbb{R}^d$, et $b, c \in \mathbb{R}^{d-1}$ les trois vecteurs décrivant la matrice A. L'algorithme* `matlab` *suivant permet d'obtenir, en fin de boucle, les valeurs des ℓ_i dans le vecteur* `b`*, celles des u_i dans* `a`*, et celles de v_i dans* `c` *(ce dernier est en fait inchangé).*

```
% Algorithme (decomposition LU tridiagonale)
for i = 1:d-1
    r = b(i)/a(i);
    a(i+1) = a(i+1) - r*c(i);
    b(i) = r;
end
```

Le coût de l'algorithme est $\mathcal{O}(d)$, ce qui est beaucoup plus faible que $\mathcal{O}(d^3)$ dans le cas général où l'on ne tire pas profit des propriétés particulières de la matrice A.

Exercice 4.

1. Montrer le théorème de Gershgorin : si λ est une valeur propre de $A \in \mathbb{R}^{d \times d}$, alors

$$\exists i \in \{1, \dots, d\}, \quad |\lambda - a_{ii}| \leqslant \sum_{j \neq i} |a_{ij}|.$$

On considérera un vecteur propre x associé à λ et i tel que $|x_i| = \max_j |x_j|$.

2. Soit la matrice

$$A = \begin{pmatrix} 7 & -1 & 2 & 0 \\ 1 & 8 & -1 & -2 \\ 0 & 1 & 7 & -1 \\ 1 & 0 & -2 & 9 \end{pmatrix}.$$

En utilisant le théorème de Gershgorin, montrer que A est inversible.

*3. Soit $D \in \mathbb{R}^{d \times d}$ une matrice diagonale et $M \in \mathbb{R}^{d \times d}$ une matrice quelconque. Montrer que, pour toute valeur propre λ de M, il existe un indice i tel que

$$|\lambda - D_{ii}| \leqslant |||M - D|||_\infty.$$

*4. Soit $A \in \mathbb{R}^{d \times d}$ diagonalisable (on note P une matrice de passage vers une base de diagonalisation), et $B \in \mathbb{R}^{d \times d}$ une matrice quelconque. Déduire de la question précédente que, pour toute valeur propre λ de B, il existe une valeur propre μ de A telle que

$$|\lambda - \mu| \leqslant \mathrm{cond}_\infty(P) |||A - B|||_\infty.$$

Correction.

1. *Soit $\lambda \in \mathbb{C}$ une valeur propre de A, et $x \in \mathbb{R}^d \backslash \{0\}$ un vecteur propre associé. On note i un indice tel que $|x_i| = \|x\|_\infty$. En développant l'égalité $(Ax)_i = \lambda x_i$, il vient*

$$\sum_{j \neq i} a_{ij} x_j = (\lambda - a_{ii}) x_i,$$

En majorant $|x_j|$ par $|x_i|$, on obtient

$$|\lambda - a_{ii}| \cdot |x_i| \leqslant |x_i| \sum_{j \neq i} |a_{ij}|.$$

Comme, en particulier, $|x_i|$ est non nul, on aboutit au résultat annoncé.

2. *Soit λ une valeur propre réelle de la matrice A. D'après le théorème de Gershgorin, cette valeur propre appartient à l'un des 4 intervalles suivants :*

$$[4, 10], \quad [4, 12], \quad [5, 9], \quad [6, 12].$$

Ainsi, λ appartient nécessairement à l'intervalle $[4, 12]$. Cet intervalle ne contenant par 0, on en déduit que λ est non nul et donc que A est inversible.

3. *Soit λ une valeur propre de la matrice M. D'après la question précédente, il existe i telle que*

$$|\lambda - M_{ii}| \leqslant \sum_{j \neq i} |M_{ij}|,$$

qui peut se récrire par inégalité triangulaire

$$|\lambda - D_{ii}| \leqslant \sum_{j \neq i} |M_{ij}| + |D_{ii} - M_{ii}|.$$

Comme $D_{ij} = 0$ pour $i \neq j$, on a encore :

$$|\lambda - D_{ii}| \leqslant \sum_{j=1}^{d} |M_{ij} - D_{ij}| \leqslant \max_{i=1}^{d} \sum_{j=1}^{d} |(M - D)_{ij}| = |||M - D|||_\infty.$$

4. *On écrit $A = PDP^{-1}$ avec D diagonale, et on pose ensuite $M = P^{-1}BP$ (qui n'est* a priori *pas diagonale). Si λ est une valeur propre de B, elle est aussi valeur propre de M, et la question précédente permet d'écrire qu'il existe i tel que*

$$|\lambda - D_{ii}| \leqslant |||M - D|||_\infty = |||P^{-1}(B - A)P|||_\infty \leqslant \mathrm{cond}_\infty(P) |||B - A|||_\infty.$$

On conclut en remarquant que D_{ii} est une valeur propre de D, donc de A !

Simulation avec `Matlab`

Exercice 5.

1. Programmer la méthode de la puissance pour le calcul d'un vecteur propre associé à la valeur propre de plus grand module d'une matrice $A \in \mathbb{R}^{d \times d}$. La tester sur la matrice

$$A = \begin{pmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 2 \end{pmatrix}.$$

On fera plusieurs tests à partir de vecteur initiaux choisis aléatoirement.

2. On considère maintenant la matrice

$$B = \begin{pmatrix} 0 & 3 & -1 \\ 4 & -1 & -1 \\ -4 & 3 & 3 \end{pmatrix}.$$

   (a) La méthode de la puissance fonctionne-t-elle avec la matrice B ? Pourquoi ?

   (b) Que fournit la méthode de la puissance appliquée à B^2 ?

3. On considère le vecteur $v_1 = (-1, \sqrt{2}, -1)^\intercal$ et on pose $\lambda_1 = 2 + \sqrt{2}$. Programmer la méthode de la puissance appliquée à la matrice

$$A_1 = A - \lambda_1 \frac{v_1 v_1^\intercal}{v_1^\intercal v_1}.$$

   Vers quelle valeur propre de A converge-t-elle ?

*4. Expliquer le phénomène. *On pourra relier les valeurs propres de A_1 à celles de A.*

Correction.

1. *Le critère d'arrêt est choisi sur l'incrément de la valeur propre, calculée comme médiane des quotients $(MX)_i/X_i$. Un nombre d'itérations maximal est aussi fixé.*

```
% Donnees
A=[2 -1 0 ; -1 2 -1 ; 0 -1 2];
B=[0 3 -1 ; 4 -1 -1 ; -4 3 3];

% Methode de la puissance
M=A; %M=B;

% Parametres
itermax=10000;
tol=1e-5;

% Iterations
x = rand(3,1);
lambda = 0;
iter = 0;
err = tol+1;
while (err>tol & iter<itermax)
    x = M*x;
    x = x/norm(x);
    lambda_new = median(M*x./x);
    err = abs(lambda-lambda_new);
    iter = iter+1;
    lambda = lambda_new;
    disp(lambda);
end
```

Pour la matrice A la méthode converge (vers la valeur propre $2 + \sqrt{2}$).

2. (a) *En revanche, pour la matrice B, la méthode ne converge pas (la « valeur propre » oscille entre des valeurs proches de -2 et -8 qui ne sont pas des valeurs propres de B). Ce phénomène est dû au fait que la matrice B possède deux valeurs propres*

de module égal au rayon spectral (4 et -4) et ne satisfait donc pas les hypothèses du théorème de convergence de la méthode de la puissance.

(b) *Pour la matrice B^2, la méthode converge. C'est conforme à la théorie car B^2 satisfait les hypothèses assurant la convergence (une seule valeur propre de module égal au rayon spectral : 16).*

3. *Le programme suivant correspond à la méthode dite* de déflation *pour les matrices symétriques.*

```
M = A - (2 + sqrt(2))*(x*x')/(x'*x);

x = rand(3,1);
lambda = 0;
iter = 0; err = tol+1;
while (err>tol) && (iter<itermax)
    x = M*x;
    x = x/norm(x);
    lambda_new = median(M*x./x);
    err = abs(lambda - lambda_new);
    iter = iter + 1;
    lambda = lambda_new
end
```

Cette méthode converge vers 2, seconde (en module) valeur propre de A.

4. *En notant $\lambda_1 = 2+\sqrt{2}$, le vecteur v_1 est vecteur propre associé à λ_1. L'itération, avant normalisation, s'écrit*

$$\tilde{x}^{(n+1)} = Ax^{(n)} - \lambda_1 \frac{v_1 v_1^T}{v_1^T v_1} x^{(n)} = Ax^{(n)} - \lambda_1 \frac{v_1^T x^{(n)}}{v_1^T v_1} v_1,$$

si bien que $\tilde{x}^{(n+1)}$ est la projection de $Ax^{(n)}$ sur l'orthogonal de v. En effet, on a déjà $\tilde{x}^{(n+1)} - Ax^{(n)}$ est parallèle à v. Par ailleurs,

$$v_1^T \tilde{x}^{(n+1)} = v_1^T A x^{(n)} - \lambda_1 v_1^T x^{(n)}$$

Or $v_1^T A = (A^T v_1)^T = (Av_1)^T = \lambda_1 v_1^T$. Donc $v_1^T \tilde{x}^{(n+1)} = 0$. Ainsi, $\tilde{x}^{(n+1)}$ appartient à l'orthogonal de v_1.

Ainsi, à chaque itération, on projette $Ax^{(n)}$ sur l'orthogonal de v_1, et c'est naturellement la plus grande valeur propre de la restriction de A à ce sous-espace qu'on approche.
Cela peut être montré en remarquant qu'il s'agit de la méthode de la puissance appliquée à la matrice

$$A_1 = A - \lambda_1 \frac{v_1 v_1^T}{v_1^T v_1},$$

qui a pour valeurs propres les mêmes que A, en remplaçant la plus grande, λ_1, par 0.

ÉCOLE CENTRALELYON

Année 2018-2019

Analyse numérique – TD 2

Intégration numérique

À préparer avant la séance de TD

Exercice 1. On considère la formule d'intégration numérique composée suivante :

$$I = \int_0^1 f(x)\,\mathrm{d}x \simeq I_N = h\sum_{i=0}^{N-1}\left[\frac{1}{4}f(ih) + \frac{3}{4}f\left(ih + \frac{2h}{3}\right)\right],$$

avec $Nh = 1$.

1. Montrer qu'il s'agit d'une méthode composée à partir d'une formule élémentaire (préciser la méthode élémentaire).
2. En déduire une estimation de l'erreur $|I - I_N|$ en fonction de h, et des dérivées de f (supposée aussi régulière que nécessaire).

Correction.

1. *La formule élémentaire associée à cette méthode composée s'écrit*

$$\int_0^1 \varphi(t)\,\mathrm{d}t \simeq \frac{\varphi(0) + 3\varphi(\frac{2}{3})}{4}.$$

2. *On étudie pour quel degré cette formule est exacte :*
   - ◇ *Pour $\varphi(t) = 1$, la formule est exacte :*

$$\int_0^1 \mathrm{d}t = 1 \quad et \quad \frac{1+3}{4} = 1.$$

   - ◇ *Pour $\varphi(t) = t$, la formule est exacte :*

$$\int_0^1 t\,\mathrm{d}t = \frac{1}{2} \quad et \quad \frac{0 + 3\times\frac{2}{3}}{4}. = \frac{1}{2}.$$

   - ◇ *Pour $\varphi(t) = t^2$, la formule est exacte :*

$$\int_0^1 t^2\,\mathrm{d}t = \frac{1}{3} \quad et \quad \frac{0 + 3\times\frac{4}{9}}{4} = \frac{1}{3}.$$

   - ◇ *Pour $\varphi(t) = t^3$, la formule n'est pas exacte :*

$$\int_0^1 t^3\,\mathrm{d}t = \frac{1}{4} \quad et \quad \frac{0 + 3\times\frac{8}{27}}{4} = \frac{2}{9}.$$

Ainsi, la formule élémentaire est exacte $\mathbb{P}_2$ (et pas $\mathbb{P}_3$). D'après le cours, on a l'estimation d'erreur suivante pour $f \in \mathscr{C}^3([0,1])$,

$$\left|\int_0^1 f(x)\,\mathrm{d}x - h\sum_{i=0}^{N-1}\left[\frac{1}{4}f(ih) + \frac{3}{4}f\left(ih + \frac{2h}{3}\right)\right]\right| \leqslant \frac{h^3}{6}\sup_{x\in[0,1]}\left|f^{(3)}(x)\right|.$$

Bonus : le programme suivant permet d'illustrer numériquement la convergence en $\mathcal{O}(h^3)$.

```
% Parametres
f=inline('exp(x)');
Sex=exp(1)-1;

liste_N=10:10:1000;
liste_Err=[];

% Boucle sur N

for N=liste_N

    % Formule composee
    h=1/N;
    S=0;
    for i=0:N-1
        S=S+h/4*(f(i*h)+3*f(i*h+2*h/3));
    end

    liste_Err=[liste_Err,abs(S-Sex)];

end

% Graphe
plot(log(liste_N),log(liste_Err))
grid on
polyfit(log(liste_N),log(liste_Err),1)
```

Exercice 2. Programmer avec `matlab` la méthode des rectangles à droite pour l'approximation de

$$I = \int_0^1 x^{-\frac{3}{4}}(1+x^2)\,\mathrm{d}x = \frac{40}{9}.$$

Reprendre les mêmes calculs avec l'intégrale obtenue après changement de variable $x = u^4$:

$$I = 4\int_0^1 (1+u^8)\,\mathrm{d}u.$$

Vérifier que les approximations obtenues à l'aide les deux méthodes pour $N = 1000$ valent respectivement 3.8335 et 4.4464.

Correction. *Le programme suivant calcule l'intégrale par la méthode des rectangles à droite, avant et après changement de variable. Le programme effectue également une comparaison des ordres de convergence (mais il est possible de comparer plus simplement les erreurs obtenues pour une même valeur de N dans les deux cas).*

```
% Parametres
I=40/9;
%lN=10.^(1:8);
lN=floor(linspace(1000,2000,10))
Err1=[];
```

```
Err2=[];

% Boucle sur N
for N=lN

    x=linspace(0,1,N+1);x=x(2:N+1);

    % Calcul avec la fonction singuliere
    I1=sum(x.^(-3/4).*(1+x.^2))/N;
    Err1=[Err1,abs(I1-I)];

    % Calcul apres changement de variable
    I2=4*sum(1+x.^8)/N;
    Err2=[Err2,abs(I2-I)];

    % Test


end

% Graphes
close all
plot(log10(lN),log10(Err1))
hold on
plot(log10(lN),log10(Err2))
grid on
xlabel('log_{10}(N)')
ylabel('log_{10}(Erreur)')

% Application numerique
N=1000;
x=linspace(0,1,N+1);x=x(2:N+1);
I1=sum(x.^(-3/4).*(1+x.^2))/N
I2=4*sum(1+x.^8)/N
```

On observe que la convergence est beaucoup plus lente dans le premier cas. On retrouve une convergence d'ordre 1 (i.e. en $\mathcal{O}(1/N)$) après changement de variable, ce qui est normal car la fonction à intégrer est de classe $\mathscr{C}^1$. Il est difficile d'estimer l'ordre de convergence pour le premier calcul, tant la convergence est lente.

À travailler pendant la séance de TD

Exercice 3. On souhaite calculer numériquement l'intégrale (pour laquelle aucune formule explicite n'existe) :

$$I = \int_1^{+\infty} \frac{e^{-x}}{x}\,\mathrm{d}x.$$

Pour ce faire, on fixe $R > 1$ et on approche I par la méthode des rectangles obtenue sur l'intervalle $[1, R]$:

$$I \simeq I_R^N = \frac{R-1}{N} \sum_{i=0}^{N-1} f(x_i),$$

où on a posé $f : x \mapsto \dfrac{e^{-x}}{x}$ et $x_i = 1 + \dfrac{i(R-1)}{N}$.

1. On pose
$$I_R = \int_1^R \frac{e^{-x}}{x}\,\mathrm{d}x.$$
Montrer que
$$|I - I_R| \leqslant \frac{e^{-R}}{R}.$$
2. Montrer que
$$\left|I_R - I_R^N\right| \leqslant \frac{(R-1)^2}{eN}.$$
3. On fixe $\varepsilon > 0$. Proposer un choix de valeurs pour R et N afin de garantir
$$\left|I - I_R^N\right| \leqslant \varepsilon.$$

Correction.

1. *Par définition,*
$$|I - I_R| = \int_R^{+\infty} \frac{e^{-x}}{x}\,\mathrm{d}x.$$
Comme $x \geqslant R$ sur l'intervalle considéré, on a
$$|I - I_R| \leqslant \frac{1}{R}\int_R^{+\infty} e^{-x}\,\mathrm{d}x = \frac{e^{-R}}{R}.$$
2. *On utilise la formule d'erreur pour la méthode des rectangles avec $a = 1$ et $b = R$:*
$$|I_R - I_R^N| \leqslant \frac{(R-1)^2}{2N}\sup_{x\in[0,R]}|f'(x)| = \frac{(R-1)^2}{eN},$$
car
$$|f'(x)| = \left|\frac{-1-x}{x^2}e^{-x}\right| \leqslant \left(\frac{1}{x} + \frac{1}{x^2}\right)e^{-x} \leqslant 2e^{-1}.$$
3. *Par inégalité triangulaire, on a*
$$|I - I_R^N| \leqslant |I - I_R| + |I_R - I_R^N| \leqslant \frac{e^{-R}}{R} + \frac{(R-1)^2}{eN}.$$
Pour assurer que $|I - I_R^N| \leqslant \varepsilon$, on choisit d'abord R tel que
$$\frac{e^{-R}}{R} \leqslant \frac{\varepsilon}{2},$$
puis N tel que
$$\frac{(R-1)^2}{eN} \leqslant \frac{\varepsilon}{2} \quad\Leftrightarrow\quad N \geqslant \frac{2(R-1)^2}{e\varepsilon}.$$

Exercice 4. On considère la formule élémentaire
$$\int_0^1 \varphi(t)\,\mathrm{d}t \simeq \frac{1}{8}\left[\varphi(0) + 3\varphi(\tfrac{1}{3}) + 3\varphi(\tfrac{2}{3}) + \varphi(1)\right].$$

Écrire la méthode composée correspondante pour le calcul de l'intégrale

$$\int_a^b f(x)\,\mathrm{d}x.$$

Combien d'évaluations de la fonction f nécessite-t-elle ? Donner une estimation de l'erreur commise par la méthode composée.

Correction. *La méthode composée s'écrit*

$$\int_a^b f(x)\,\mathrm{d}x \simeq I_N = \frac{h}{8}\sum_{i=0}^{N-1}\left[f(x_i) + 3f(x_i+\tfrac{h}{3}) + 3f(\tfrac{2h}{3}) + f(x_{i+1})\right].$$

A priori, *cette méthode nécessite $4N$ évaluations de la fonction, mais si on ne recalcule par $f(x_i)$ pour $i \geqslant 1$ car il a été calculé dans l'intervalle précédent, le nombre d'évaluations devient $3N+1$.*
Un calcul direct comme dans l'exercice 1 montre que le modèle élémentaire est exact $\mathbb{P}_3$ (mais pas $\mathbb{P}_4$). On a donc l'estimation d'erreur suivante :

$$|I - I_N| \leqslant \frac{(b-a)^5}{24N^4}\sup_{x\in[a,b]}|f^{(4)}(x)|.$$

Simulation avec `Matlab`

Exercice 5. On considère une formule de quadrature élémentaire de la forme

$$\int_0^1 \varphi(t)\,\mathrm{d}t \simeq \sum_{q=0}^{k} w_q\varphi(t_q),$$

avec $t_q = q/k$. Cette formule est exacte pour les polynômes de degré au plus k si et seulement si

$$\forall i = 0,1,\dots k, \quad \sum_{q=0}^{k} w_q t_q^i = \frac{1}{i+1}.$$

1. Programmer la résolution de ce système linéaire.
2. Vérifier que le cas $k=3$ correspond à la méthode de l'exercice 4.
3. Programmer la méthode composée correspondant à $k=3$, et illustrer numériquement son ordre de convergence.

Correction. *Le programme suivant calcule les poids de quadrature pour ces méthodes (dites de Newton-Cotes) pour $k = 1,2,\dots,9$, et trace les graphes de convergence (vis-à-vis de N) sur un cas-test. Pour les méthodes d'ordre élevé, on observe une saturation, due à la précision-machine. Par ailleurs, on peut noter que, pour $n=8$, la méthode fait apparaître un poids négatif (cela peut entraîner des problèmes de stabilité, mais ils ne sont pas visibles ici).*

```
clear
close all
% Cas test
f=inline('exp(x)');
a=0;
```

```
b=1;
Iex=exp(1)-1;


% Parametres
ToutesErreurs=[];
str_legend=[];
listeN=1:5:100

% Boucles sur k et N

for k=1:9 % (k+1) est le nombre de points de la methode
   elementaire

    disp(['k=',num2str(k)])
    % Calcul des poids
    t=[0:k]/k;
    V=[];
    for i=1:k+1
        V=[V;t.^(i-1)];
    end
    rhs=1./[1:k+1]';
    w=V\rhs;

    % Programmation de la methode

    Erreurs=[];
    for N=listeN
        disp(['...N=',num2str(N)]);
        h=(b-a)/N;
        I=0;
        for i=1:N
            for q=1:k+1
                I=I+w(q)*f(a+h*(i-1)+h*(q-1)/k);
            end
        end
        I=I*(b-a)/N;

        % Erreur relative
        Erreurs=[Erreurs,abs(I-Iex)/Iex];
    end

    ToutesErreurs=[ToutesErreurs;Erreurs];
    str_legend=[str_legend;['k=',num2str(k)]];

end

plot(log(listeN),log10(ToutesErreurs),'*-');
grid on
```

```
54 legend(str_legend)
55 xlabel('log_{10}(N)')
56 ylabel('log_{10}(Erreur)')
```

ÉCOLE CENTRALELYON

Année 2018-2019

Analyse numérique – TD 3

Optimisation numérique

Version du 28 janvier 2019

À préparer avant la séance de TD

Exercice 1. On considère la fonction $f : \mathbb{R}^3 \to \mathbb{R}$ définie par

$$f(x, y, z) = 1 + x^2 + |y|^3 + z^2 + \sin z.$$

1. Vérifier que f est de classe $\mathscr{C}^2$ et qu'elle admet un unique point de minimum global.
2. Expliciter les itérations de la méthode de Newton pour la minimisation de f.
3. Mettre en œuvre la méthode précédente avec `Matlab`.
4. Vérifier que les approximations du point de minimum et du minimum de f obtenues avec 20 itérations de la méthode de Newton en partant du vecteur $(1, 1, 1)$ valent respectivement $[0; 0; -0.4502]$ et 0.7675.

*5. Observer la vitesse de convergence de chaque composante. Commenter.

Correction.

1. *La fonction $y \mapsto |y|^3$ est $\mathscr{C}^2$ sur $\mathbb{R}$ (sa dérivée seconde est $y \mapsto 6|y|$) donc f est $\mathscr{C}^2$. On peut raisonner composante par composante : f est minimale si x^2 et $|y|^3$ et $z^2 + \sin z$ sont minimales ce qui impose $x = y = 0$. De plus $h(z) = z^2 + \sin z$ a pour dérivée $h'(z) = 2z + \cos z$ et dérivée seconde $h''(z) = 2 - \sin z > 0$. h est donc strictement (même fortement) convexe et coercive donc admet un unique point de minimum z^*.*
2. *Par définition :*

$$\begin{bmatrix} x_{n+1} \\ y_{n+1} \\ z_{n+1} \end{bmatrix} = \begin{bmatrix} x_n \\ y_n \\ z_n \end{bmatrix} - \big(Hf(x_n, y_n, z_n)\big)^{-1} \nabla f(x_n, y_n, z_n)$$

avec $\nabla f(x, y, z) = \begin{bmatrix} 2x \\ 3y|y| \\ 2z + \cos z \end{bmatrix}$ *et* $Hf(x, y, z) = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 6|y| & 0 \\ 0 & 0 & 2 - \sin z \end{bmatrix}$. *Ainsi,*

$$\big(Hf(x_n, y_n, z_n)\big)^{-1} \nabla f(x_n, y_n, z_n) = \begin{bmatrix} x_n \\ \frac{y_n}{2} \\ \frac{2z_n + \cos z_n}{2 - \sin z_n} \end{bmatrix}.$$

Finalement,

$$\begin{cases} x_{n+1} = 0 \\ y_{n+1} = \dfrac{y_n}{2}, \\ z_{n+1} = z_n - \dfrac{2z_n + \cos z_n}{2 - \sin z_n}. \end{cases}$$

3.

```
clear
close all

f = @(x) 1 + x(1)^2 + abs(x(2))^3 + x(3)^2 + sin(x(3));

x = [1 ; 1 ; 1];

Niter = 20;

for i = 1 : Niter
   grad = [2*x(1) ; 3*x(2)*abs(x(2)) ; 2*x(3)+cos(x(3))];
   Hessienne = [2 0 0 ; 0 6*abs(x(2)) 0 ; 0 0 2-sin(x(3))
      ];
   x = x - Hessienne\grad;
end

x
f(x)
```

4. *Le point de minimum est* $\boxed{[0;0;-0.4502]}$ *e tel minimum de* f *vaut* $\boxed{0.7675}$.

5. *L'erreur sur* x *est nulle dès la premiere itération (ce qui est attendu car la fonction à annuler – première composante du gradient – est linéaire). L'erreur sur* y *est géométrique de raison* $\frac{1}{2}$ *(ce qui est plus lent que ce qui est attendu – on peut montrer que cela est dû au fait que la racine recherchée est multiple). L'erreur sur* z *est quadratique :* $e_z(n) = \alpha\beta^{2^n}$ *avec* $\alpha > 0$ *et* $\beta \in]0,1[$. *On voit numériquement qu'elle est mieux que géométrique (graphe concave de* $(n, \log(e_z(n)))$. *On peut le vérifier en traçant* $\ln(-\ln e_z(n)) \sim n \ln 2$ *attention à la saturation d'erreur.*

```
clear
close all

x = [1 ; 1 ; 1];
Niter = 20;
xexact = [0;0;-0.450183611294874];
erreur = zeros(3,Niter);

for i = 1 : Niter
   grad = [2*x(1) ; 3*x(2)*abs(x(2)) ; 2*x(3)+cos(x(3))];
   Hessienne = [2 0 0 ; 0 6*abs(x(2)) 0 ; 0 0 2-sin(x(3))
      ];
   x = x - Hessienne\grad;
   erreur(:,i) = abs(x-xexact);
end

% Erreur sur x
format shortEng
max(erreur(1,:))
format short

```

```
% Erreur sur y
plot(1:Niter,log10(erreur(2,:)),'-o')
title('Erreur sur y')
a = polyfit(1:Niter,log10(erreur(2,:)),1);
disp('Convergence geometrique de raison')
10^(a(1))

% Erreur sur z
figure
plot(1:Niter,log10(erreur(3,:)),'-o')
title('Erreur sur z en log')
figure
plot(1:Niter,log(-log(erreur(3,:))),'-o')
a = polyfit(2:5,log(-log(erreur(3,2:5))),1);
title('Erreur sur z en log(-log)')
disp('Convergence quadratique :')
exp(a(1))
```

À travailler pendant la séance de TD

Exercice 2. On considère la fonction $f(x,y) = 1 + x^2 + |y|^3$ et la suite $(x_n, y_n)_{n\in\mathbb{N}}$ de $\mathbb{R}^2$ définie par la méthode du gradient à pas fixe

$$\begin{cases} (x_{n+1}, y_{n+1}) = (x_n, y_n) - \rho \nabla f(x_n, y_n), \\ \qquad (x_0, y_0) \in \mathbb{R}^2. \end{cases}$$

1. Montrer que $x_n = (1-2\rho)^n x_0$ pour tout $n \in \mathbb{N}$.
2. On suppose $y_0 > 0$ et $\rho < \min\left(\frac{1}{3y_0}, 1\right)$. Montrer que (x_n, y_n) tend vers $(0,0)$.
   On montrera que la suite (y_n) est strictement positive et décroissante.
3. Montrer par l'absurde que la convergence de (y_n) n'est pas géométrique. Quelle hypothèse sur f manque-t-il pour que le théorème du cours sur la vitesse de convergence fonctionne ?

Correction.

1. *Immediat.*
2. *On a $y_{n+1} = (1-3\rho|y_n|)y_n$. Par hypothèse, $0 < y_0 < \frac{1}{3\rho}$ donc $(1-3\rho y_0) \in]0,1[$ ainsi $y_1 \in]0, y_0[$. Par récurrence, si $y_0, \dots, y_n$ est strictement positive décroissante, alors $0 < y_n \leq y_0 < \frac{1}{3\rho}$ donc $1 - 3\rho|y_n| \in]0,1[$ donc $y_{n+1} \in]0, y_n[$ et donc $y_0, \dots, y_{n+1}$ est strictement positive décroissante . Ainsi (y_n) est décroissante minorée donc convergente vers y^*. En passant à la limite, $y^* = (1-3\rho|y^*|)y^*$ donc $y^* = 0$. D'après la question 1, $|x_n| \leq |1-2\rho|^n |x_0|$. comme $\rho \in]0,1[$, $|1-2\rho| < 1$ ce qui garantit la convergence de (x_n) vers 0.*
3. *Remarquons que $\frac{y_{n+1}}{y_n} \xrightarrow{n\infty} 1$ On va montrer que ce n'est pas compatible avec une convergence géométrique. Supposons que la convergence soit géométrique. Par définition,*

$$\exists \alpha \in]0,1[, \quad \exists C > 0, \quad \forall n \in \mathbb{N}, \quad y_n \leq C\alpha^n.$$

Considérons $\beta \in]\alpha, 1[$, il existe un rang $N \in \mathbb{N}$ tel que pour tout $n \geq N$, $\frac{y_{n+1}}{y_n} \geq \beta$ donc $y_{n+1} \geq \beta y_n$. Par une récurrence immédiate, on a

$$\forall n \geq N, \quad y_n \geq y_N \beta^{n-N}.$$

Pour conclure, il suffit de choisir un n tel que $y_N \beta^{n-N} > C\alpha^n$ c'est-à-dire tel que

$$\left(\frac{\beta}{\alpha}\right)^n > \frac{C}{y_N \beta^N}$$

ce qui est possible car $\frac{\beta}{\alpha} > 1$. Ainsi, $y_n > C\alpha^n$, ce qui est absurde.
Le théorème du cours fonctionne pour les fonctions α-convexes. Or ici f ne l'est pas. En effet les valeurs propres de sa hessienne sont 2 et $6|y|$. Cette dernière tend vers 0 quand y tend vers 0. La vitesse de convergence n'est pas garantie dans ce cas.

Exercice 3. On considère le problème d'optimisation

$$\max\{(Ax|x) \ ; \ x \in \mathcal{B}\},$$

où A est une matrice symétrique réelle définie positive et $\mathcal{B} = \{x \in \mathbb{R}^d \ ; \|x\|_2 \leqslant 1\}$.

1. Donner l'expression de la projection sur $\mathcal{B}$.
2. Écrire l'itération donnée par la méthode du gradient projeté appliquée à ce problème.
3. On suppose $\|x^{(0)}\|_2 \geqslant 1$. Quelle méthode vue au chapitre 1 reconnaît-on ?

*4. Quelle est est la limite de $(Ax^{(n)}|x^{(n)})$?

Correction.

1. *$\pi_{\mathcal{B}}(x) = x\chi_{\mathcal{B}}(x) + \frac{x}{\|x\|_2}\chi_{\mathbb{R}^d \setminus \mathcal{B}}(x)$.*
2. *Le gradient de $f(x) = -(Ax|x)$ est $\nabla f(x) = -2Ax$ ainsi l'algorithme s'écrit*

$$\begin{cases} x^{(0)} \in \mathbb{R}^d, \\ \tilde{x} = x^{(n)} + 2\rho A x^{(n)}, \\ x^{(n+1)} = \pi_{\mathcal{B}}(\tilde{x}). \end{cases}$$

La fonction f n'étant pas convexe, on ne peut pas en déduire directement la convergence de la méthode. Il faut attendre la question suivante.

3. *La matrice A est définie positive. On note ses valeurs propres $0 < \lambda_1 \leq \cdots \leq \lambda_d$. On remarque une ressemblance avec la méthode de la puissance. Pour que ce soit exactement identique, il faut (et suffit) que $\|\tilde{x}\|_2 \geqslant 1$ à chaque étape (et on a alors $\|x^{(n)}\|_2 = 1$ pour tout n). Or, on peut écrire*

$$x^{(n)} = (I + 2\rho A)^{-1}\tilde{x}, \quad \text{donc} \quad \|x^{(n)}\|_2 \leq |||(I + 2\rho A)^{-1}|||_2 \|\tilde{x}\|_2.$$

Mais

$$|||(I + 2\rho A)^{-1}|||_2 = \frac{1}{1 + 2\rho\lambda_1} < 1.$$

Donc $\|x^{(n)}\|_2 \leq \|\tilde{x}\|_2$, ce qui montre bien que $\|\tilde{x}\|_2 \geqslant 1$.
L'algorithme s'écrit donc

$$\tilde{x} = (I + 2\rho A)x^{(n)},$$
$$x^{(n+1)} = \frac{\tilde{x}}{\|\tilde{x}\|_2}.$$

C'est la méthode de la puissance sur la matrice $I + 2\rho A$.

La convergence est assurée d'après le chapitre 1 car sa valeur propre de module maximal est unique (c'est $1 + 2\rho\lambda_d$).

4. *D'après la question précédente, x_n converge vers x^*, un vecteur propre normé associé à la valeur propre $1 + 2\rho\lambda_d$. Ainsi, $(Ax^{(n)}|x^{(n)})$ converge vers λ_d.*

Simulation avec `Matlab`

Exercice 4. Nous allons résoudre le problème du cable pesant grâce à la méthode du gradient pénalisé. On cherche $(\mathbf{x}, \mathbf{y}) \in \mathbb{R}^{2N}$ solution de

$$\min_{(\mathbf{x},\mathbf{y})\in K} E_p(\mathbf{x}, \mathbf{y})$$

avec $E_p(\mathbf{x}, \mathbf{y}) = \dfrac{1}{N+1}\displaystyle\sum_{i=1}^{N} y_i$ et $K = \{(\mathbf{x}, \mathbf{y}) \in \mathbb{R}^{2N}, \varphi_i(\mathbf{x}, \mathbf{y}) = 0,\ i = 0 \dots N\}$. Les contraintes de longueur sont données par

$$\varphi_i(\mathbf{x}, \mathbf{y}) = \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2} - \frac{L}{(N+1)}$$

où l'on suppose que $(x_0, y_0) = (0, 1)$ et $(x_{N+1}, y_{N+1}) = (1, 3/2)$. On notera $\mathbf{u} = (\mathbf{x}, \mathbf{y}) \in \mathbb{R}^{2N}$.

1. Écrire une fonction `[val,grad] = energie(u)` qui renvoie l'énergie potentielle du cable ainsi que son gradient. Télécharger le fichier penalisation.m (sur la plateforme pédagogique) qui définit la fonction

   `[val,grad] = penalisation(u)`

   qui renvoie la valeur de la fonction $\beta(\mathbf{u}) = \frac{1}{2}\sum_{i=0}^{N} \varphi_i^2(\mathbf{u})$ ainsi que son gradient.
2. Mettre en œuvre la méthode du gradient à pas fixe pour résoudre le problème pénalisé

   $$\min_{\mathbf{u}\in\mathbb{R}^{2N}} E_p(\mathbf{u}) + \frac{1}{\varepsilon}\beta(\mathbf{u}). \tag{2}$$

   On prendra pour $\mathbf{u}_0$ un vecteur aléatoire, $L = \frac{3}{2}$, $N = 20$ et on testera $\varepsilon = 0.1$, $\varepsilon = 10^{-2}$ et $\varepsilon = 10^{-3}$. On prendra toujours $\rho = \varepsilon/3$. L'algorithme s'arrêtera dès que $\left\|\nabla E_p(\mathbf{u}) + \frac{1}{\varepsilon}\nabla\beta(\mathbf{u})\right\|_\infty \leq 10^{-3}$. À la fin de l'algorithme, on récupérera la position `(x,y)` du cable on on l'affichera avec la commande `plot(x,y,'-o')`.
3. Écrire une fonction `L = longueur(x,y)` qui calcule et renvoie la longueur du cable $\mathbf{u}$. Comment cette longueur évolue-t-elle quand on diminue ε ?

*4. On ajoute à notre problème la contrainte supplémentaire

$$y_i \geq x_i + \frac{1}{4} \quad i = 1 \dots N.$$

Donner la projection $\pi : \mathbb{R}^2 \to K$ sur l'ensemble

$$D_1 = \left\{(x, y) \in \mathbb{R}^2,\ y \geq x + \frac{1}{4}\right\}.$$

En déduire une fonction matlab `[xx,yy] = projection(x,y)` qui projette le cable $(\mathbf{x}, \mathbf{y})$ sur l'ensemble

$$D_N = \left\{ (\mathbf{x}, \mathbf{y}) \in \mathbb{R}^{2N},\ y_i \geq x_i + \frac{1}{4},\ i = 1 \dots N \right\}.$$

Résoudre enfin le problème (2) dans l'ensemble admissible D_N.

Correction.

```
function [val,grad] = energie(u)
N = length(u)/2;
y = u(N+1:2*N);
val = sum(y)/(N+1);
grad = [zeros(N,1) ; ones(N,1)/(N+1)];
```

Le mieux est que les élèves mettent en œuvre l'algo dans un script simple avec un nombre faible d'iteration d'abord en un affichage dynamique du cable.

```
clear; close all;

% Parmetres
N = 20;
eps = 0.01;
Nitermax = 500;
rho = eps/3;
u = rand(2*N,1);

% Iterations
for i = 1 : Nitermax
   [vale,grade] = energie(u);
   [valp,gradp] = penalisation(u);
   g = grade+1/eps*gradp;

    u = u - rho*g;

    x = [0 ; u(1:N) ; 1];
    y = [1 ; u(N+1:2*N) ; 3/2];


   % Affichage dynamique
   plot(x,y,'-o');
   title(['i = ' num2str(i)])
   pause(0.01);
end
```

Bien faire enlever les affichages graphiques dans la fonction cable.

```
function [x,y,Niter] = cable(N,eps)
Nitermax = 100000;

u = rand(2*N,1);
```

```
rho = eps/3;
for i = 1 : Nitermax
   [~,grade] = energie(u);
   [~,gradp] = penalisation(u);
   g = grade+1/eps*gradp;

   u = u - rho*g;

   if max(abs(g)) < 1e-3
        break
   end
end

Niter = i;
x = [0 ; u(1:N) ; 1];
y = [1 ; u(N+1:2*N) ; 3/2];
```

```
function L = longueur(x,y)

 L = sum(sqrt(diff(x).^2 + diff(y).^2));
```

Voici un script qui permet de mesurer la longueur du cable pour différentes valeur de ε et de les afficher. On observera la convergence vers $3/2$.

```
clear; close all;

N = 20;
eps = 1;
erreur = [];
epslist = logspace(-1,-3,11);
for eps = epslist
    [x,y,Niter] = cable(N,eps);
    L = longueur(x,y);
    disp(['L = ' num2str(L)])
    plot(x,y,'-o')

    hold on
    erreur = [erreur ,abs(L-3/2)];
end
title(['Cable pour $\log \varepsilon\in [$' ...
    num2str(log10(epslist)) ']'],'Interpreter','latex')

figure
plot(log10(epslist),log10(erreur),'-o')
a = polyfit(log10(epslist),log10(erreur),1);
disp([' Ordre = ' num2str(a(1))])
xlabel('$\log(\varepsilon)$','Interpreter','latex')
ylabel('$\log|L_{\varepsilon} - L|$','Interpreter','latex')
```

C'est une projection orthogonale sur un demi espace. Si $y \geq x + 1/4$, $\pi(x,y) = (x,y)$, et

sinon,

$$\pi(x,y) = \frac{1}{2}\big((x,y-1/4)|(1,1)\big)(1,1) + (0,1/4).$$

$$\pi(x,y) = \left(\frac{x}{2} + \frac{y}{2} - \frac{1}{8},\ \frac{x}{2} + \frac{y}{2} + \frac{1}{8}\right).$$

Faire dessiner le cable solution et la droite d'équation $y = x + 1/4$.

```
function [xx,yy] = projection(x,y)
xx = x.*(y>=x+1/4) + (x/2+y/2-1/8).*(y<x+1/4);
yy = y.*(y>=x+1/4) + (x/2+y/2+1/8).*(y<x+1/4);
```

ÉCOLE CENTRALELYON

Année 2018-2019

Analyse numérique – TD 4

Approximation des EDO

À préparer avant la séance de TD

Exercice 1. Un substance chimique y est libérée dans la nature à partir de $t = 0$ à un débit de e^{-t^2}. Celle-ci se dégrade très rapidement en une seconde substance z qui est lentement dissipée dans l'environnement. Les concentrations de ces substances vérifient le système d'équations différentielles

$$\begin{cases} y'(t) = -50y(t) + e^{-t^2}, \\ z'(t) = 50y(t) - 2z^2(t), \\ y(0) = z(0) = 0. \end{cases}$$

La solution (y, z) de ce problème est globale, positive et bornée. On s'intéresse à la quantité résiduelle de substance z à long terme ($t = 1000$).

1. Mettre en œuvre avec `Matlab` la méthode d'Euler pour résoudre ce problème sur l'intervalle de temps $[0, 5]$. Tracer les solutions pour $N = 125$ et $N = 250$.
2. Dans le cas où $N = 250$, vérifier que la valeur approchée de $z(5)$ vaut environ $0,10778$.

*3. Montrer que la méthode d'Euler implicite conduit à définir une suite $(Y_n, Z_n)_{n\in\mathbb{N}}$ par

$$\begin{cases} Y_{n+1} = \dfrac{Y_n + he^{-t_{n+1}^2}}{1 + 50h}, \\ Z_{n+1} = \dfrac{\sqrt{1 + 8hZ_n + 400h^2Y_{n+1}} - 1}{4h}, \\ Y_0 = Z_0 = 0. \end{cases}$$

Mettre en œuvre cette méthode sur l'intervalle $[0, 5]$ et comparer les résultats avec la méthode précédente pour un pas $h = 0.04$ puis $h = 0.01$. Comparer avec la méthode d'Euler explicite.

*4. Donner l'approximation de $z(1000)$ en utilisant la méthode d'Euler implicite avec $h = 0.05$.

Correction.

1.

```
function up=f(t,u)
up=[-50*u(1) + exp(-t^2);
   50*u(1) - 2*u(2)^2];
```

```
function [u,t] = EulerExp(Nh,T)
h=T/Nh;
```

```
t = h*(0:Nh);

u = zeros(2,Nh+1);

for n = 1 : Nh
   u(:,n+1)=u(:,n)+h*f(t(n),u(:,n));
end
```

```
clear; close all;

% parametres
T = 5;
Nh = 250;

% Euler explicite
[u,t] = EulerExp(Nh,T);

% Plots
subplot(1,2,1)
plot(t,u(1,:))
title('y(t) Euler exp')
subplot(1,2,2)
plot(t,u(2,:))
title('z(t) Euler exp')

disp(['z(5) = ' num2str(u(2,Nh+1))])
```

2. On trouve $\boxed{z(5) \approx 0,10778}$.

3. Pour Euler implicite on a par définition :

$$\begin{cases} Y_{n+1} = Y_n - 50hY_{n+1} + he^{-t_{n+1}^2}, \\ Z_{n+1} = Z_n + 50hY_{n+1} - 2hZ_{n+1}^2 \\ Y_0 = Z_0 = 0. \end{cases}$$

$$\Leftrightarrow \begin{cases} Y_{n+1} = \dfrac{Y_n + he^{-h^2(n+1)^2}}{1+50h}, \\ 2hZ_{n+1}^2 + Z_{n+1} - (Z_n + 50hY_{n+1}) = 0, \\ Y_0 = Z_0 = 0. \end{cases}$$

Afin de rendre la seconde equation explicite en Z_{n+1} *on cherche les racines de* $2hX^2+X-\alpha$ *:* $\Delta = 1+8h\alpha$ *(*> 0 *par hypothèse de récurrence) donc* $X = \frac{\pm\sqrt{1+8h\alpha}-1}{4h}$ *; On garde alors la racine positive.* $Z_{n+1} = \frac{\sqrt{1+8h\alpha}-1}{4h}$

```
function [u,t] = EulerImp(Nh,T)
h=T/Nh;
t = h*(0:Nh);

y = zeros(1,Nh+1);
z = zeros(1,Nh+1);
```

```
for i = 1 : Nh
   y(i+1) = (y(i) + h*exp(-t(i+1)^2))/(1+h*50);
   z(i+1) = (sqrt(1+8*h*z(i)+400*h^2*y(i+1)) - 1) / (4*h);
end

u=[y;z];
```

Script de mise en œuvre comparative :

```
clear; close all;

% parametres
T = 5;


h = 0.04;
Nh = T/h;

% Euler explicite
[uexp,t] = EulerExp(Nh,T);
% Euler implicite
[uimp,t] = EulerImp(Nh,T);

subplot(2,2,1)
plot(t,uexp(1,:),t,uimp(1,:))
title('y(t) avec h = 0.04')
legend('Explicite','Implicite')
subplot(2,2,2)
plot(t,uexp(2,:),t,uimp(2,:))
title('z(t) Euler avec h = 0.04')
legend('Explicite','Implicite')

h = 0.01;
Nh = T/h;

% Euler explicite
[uexp,t] = EulerExp(Nh,T);
% Euler implicite
[uimp,t] = EulerImp(Nh,T);

subplot(2,2,3)
plot(t,uexp(1,:),t,uimp(1,:))
title('y(t) avec h = 0.01')
legend('Explicite','Implicite')
subplot(2,2,4)
plot(t,uexp(2,:),t,uimp(2,:))
title('z(t) Euler avec h = 0.01')
legend('Explicite','Implicite')
```

4. Voici un script pour l'évaluation de $z(1000)$:

```
clear; close all;

% parametres
T = 5;
Nh = 125;

% Euler explicite


[u,t] = EulerExp(Nh,T);
subplot(2,2,1)
plot(t,u(1,:))
title('y(t) Euler exp')
subplot(2,2,2)
plot(t,u(2,:))
title('z(t) Euler exp')

% Euler implicite
subplot(2,1,2)
Nh=1000;
[u,t] = EulerImp(Nh,T);
subplot(2,2,3)
plot(t,u(1,:))
title('y(t) Euler imp')
subplot(2,2,4)
plot(t,u(2,:))
title('z(t) Euler imp')
```

La valeur numérique demandée pour $z(1000)$ est $\boxed{5.0034.10^{-4}}$.

À travailler pendant la séance de TD

Exercice 2. On considère un problème de Cauchy

$$(C) : \begin{cases} u'(t) = f(t, u(t)), & \text{sur } [0, T], \\ u(0) = u_0, \end{cases}$$

où $f : [0, T] \times \mathbb{R} \to \mathbb{R}$ est de classe $\mathscr{C}^2$ et il existe une constante $L > 0$ telle que

$$\forall t \in [0, T], \quad \forall v, w \in \mathbb{R}, \quad |f(t, v) - f(t, w)| \leqslant L|v - w|.$$

On considère la méthode donnée par

$$\begin{cases} U_{n+1} = U_n + \dfrac{h}{2}\Big[3f(t_n, U_n) - f(t_{n-1}, U_{n-1})\Big] & \forall n \geqslant 1, \\ U_0 = u_0, \\ U_1 = u_0 + hf(t_0, U_0). \end{cases}$$

1. Justifier que (C) admet une unique solution globale $u : [0, T] \to \mathbb{R}$ de classe $\mathscr{C}^3$.

2. Définir l'erreur de consistance ε_n^h de la méthode et montrer que

$$|\varepsilon_n^h| \leqslant Ch^2 \sup_{t\in[0,T]} |u'''(t)|,$$

où C est une constante que l'on précisera.

3. Montrer que la méthode est stable.
On pourra poser $E_n = \max(|U_n - V_n|, |U_{n-1}, V_{n-1}|)$.

4. En déduire que la méthode est convergente.

Correction. *N.B. la méthode proposée ici est à deux pas, car U_{n+1} dépend de U_n mais aussi de U_{n-1}. Les résultats vus en cours pour les méthodes à un pas ne peuvent pas s'appliquer directement ici. Il s'agit d'adapter les démonstrations faite en cours ici. En fait, une bonne compréhension de l'analyse de la méthode d'Euler permet de comprendre aisément les preuves pour cette nouvelle méthode.*

1. *Comme f est globalement lipschitzienne en u, le théorème de Cauchy-Lipschitz global s'applique, assurant l'existance et l'unicité de $u \in \mathscr{C}^1([0,T],\mathbb{R})$.*
Comme $u'(t) = f(t,u(t))$ et que $f \in \mathscr{C}^2$, $u \in \mathscr{C}^1$, on en déduit $u \in \mathscr{C}^2$. Mais, par suite, f et u sont $\mathscr{C}^2$ donc u est $\mathscr{C}^3$.

2. *L'erreur de consistance est*

$$\varepsilon_n^h = \frac{u(t_{n+1}) - u(t_n)}{h} - \frac{1}{2}\Big(3f\big(t_n, u(t_n)\big) - f\big(t_{n-1}, u(t_{n-1})\big)\Big)$$

$$= \frac{u(t_{n+1}) - u(t_n)}{h} - \frac{1}{2}\big(3u'(t_n) - u'(t_{n-1})\big)$$

En utilisant des développements de Taylor, il existe θ_n^1 et θ_n^2 dans l'intervalle $[t_n, t_{n+1}]$ tels que

$$u(t_{n+1}) = u(t_n) + hu'(t_n) + \frac{h^2}{2}u''(t_n) + \frac{h^3}{6}u'''(\theta_n^1),$$

$$u'(t_{n-1}) = u'(t_n) - hu''(t_n) + \frac{h^2}{2}u'''(\theta_n^2).$$

On en déduit que

$$\varepsilon_n^h = \frac{h^2}{6}u'''(\theta_n^1) + \frac{h^2}{4}u'''(\theta_n^2),$$

puis la majoration

$$|\varepsilon_n^h| \leqslant \frac{5h^2}{12} \sup_{t\in[0,T]} |u'''(t)|.$$

Le schéma est consistant d'ordre 2.

3. *Si V_n est solution d'un schéma perturbé,*

$$\begin{cases} V_{n+1} = V_n + \frac{h}{2}\Big[3f(t_n, V_n) - f(t_{n-1}, V_{n-1})\Big] + \mu_n \quad \forall n \geqslant 1, \\ V_1 = u_0 + hf(t_0, u_0), \\ V_0 = u_0, \end{cases}$$

alors

$$\begin{aligned}|V_{n+1}-U_{n+1}| &\leq |V_n-U_n| + \frac{3h}{2}|f(t_n,U_n)-f(t_n,V_n)| \\ &\quad + \frac{h}{2}|f(t_{n-1},U_{n-1})-f(t_{n-1},V_{n-1})| + |\mu_n| \\ &\leqslant \Big(1+\frac{3hL}{2}\Big)|U_n-V_n| + \frac{hL}{2}|U_{n-1}-V_{n-1}| + |\mu_n| \\ &\leqslant \Big(1+2hL\Big)E_n + |\mu_n|,\end{aligned}$$

où on a noté $E_n = \max(|U_n - V_n|, |U_{n-1}, V_{n-1}|)$. *On en déduit immédiatement que*

$$E_{n+1} \leqslant (1+2hL)E_n + |\mu_n|.$$

On retrouve une inégalité du même type que celle vue en cours pour la méthode d'Euler (en remplaçant L *par* $2L$*). La conclusion s'écrit*

$$E_n \leqslant (1+2hL)^n \left(E_0 + \sum_{k=1}^{n-1}|\mu_k|\right) \leqslant (1+2hL)^{N_h}\left(E_0 + \sum_{k=1}^{N_h-1}|\mu_k|\right),$$

qui fournit finalement

$$\begin{aligned}\forall n \geqslant 1, \quad |U_n - V_n| &\leqslant e^{2LT}\left(\max\left(|U_0-V_0|, |U_1-V_1|\right) + \sum_{n=1}^{N_h-1}|\mu_n|\right) \\ &\leqslant e^{2LT}\left(|U_1-V_1| + \sum_{n=1}^{N_h-1}|\mu_n|\right).\end{aligned}$$

4. *Pour la convergence, on considère la suite* $V_n = u(t_n)$. *Alors, par définition, on a*

$$\mu_n = h\varepsilon_n^h.$$

L'estimation de stabilité prouvée à la question précédente assure donc que

$$\forall n \geqslant 1, \quad |U_n - V_n| \leqslant e^{2LT}\left(|U_1 - u(t_1)| + h\sum_{n=1}^{N_h-1}|\varepsilon_n^h|\right).$$

Pour estimer le terme $U_1 - u(t_1)$, *il suffit d'écrire :*

$$U_1 - u(t_1) = u_0 + hf(t_0,u_0) = u(0) + hu'(0) - u(h) = \frac{h^2}{2}u''(\theta),$$

avec $\theta \in [0,h]$. *En exploitant l'estimation de consistance (question 2.), on obtient donc*

$$\forall n \geqslant 1, \quad |U_n - u(t_n)| \leqslant e^{2LT}h^2\left(\frac{1}{2}\sup_{t\in[0,T]}|u''(t)| + \frac{5T}{12}\sup_{t\in[0,T]}|u'''(t)|\right),$$

qui prouve la convergence à l'ordre 2 de la méthode.

ÉCOLE CENTRALELYON

Année 2018-2019

Analyse numérique – TD 5

Résolution numérique de l'équation de Laplace

À préparer avant la séance de TD

Exercice 1. On considère le problème aux limites suivant

$$\begin{cases} -u''(x) + c(x)u(x) = f(x), \ x \in]0,1[, \\ u(0) = 0, \quad u(1) = -1, \end{cases}$$

avec $c(x) = x$ et $f(x) = (x^2 + \pi^2 x)\cos(\pi x) + 2\pi \sin(\pi x)$. La solution exacte est donnée par $u(x) = x\cos(\pi x)$. Adapter la méthode des différences finies vue en cours sur cet exemple. Vérifier que l'approximation de l'intégrale

$$\int_0^1 u(x)\,\mathrm{d}x,$$

obtenue à l'aide de la méthode des rectangles à gauche pour $N = 100$ vaut -0.19765. On pourra également vérifier que la vitesse de convergence est celle prouvée en cours.

Correction. *Par rapport à ce qui a été vu en cours, la différence réside dans la présence du terme $c(x)u(x)$, qui se traduit par l'ajout de termes diagonaux $c(x_i)$ sur la diagonale de la matrice.*
Le système linéaire à résoudre s'écrit donc $AU = F$, avec

$$A = \frac{1}{h^2}\begin{pmatrix} 2 & -1 & & & \\ -1 & \ddots & \ddots & & \\ & \ddots & \ddots & -1 & \\ & & -1 & 2 & \end{pmatrix} + \begin{pmatrix} c(x_1) & & & \\ & \ddots & & \\ & & & c(x_N) \end{pmatrix}$$

et

$$U = \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_N \end{pmatrix}, \quad F = \begin{pmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ f(x_{N-1}) \\ f(x_N) - 1/h^2 \end{pmatrix}.$$

Le programme suivant effectue la résolution, et la comparaison avec la solution exacte.

```
clear
close all

% Cas-test
c=inline('x');
f=inline('(x.^2+pi^2*x).*cos(pi*x)+2*pi*sin(pi*x)');
uex=inline('x.*cos(pi*x)');
```

```
8
9  % Parametres
10 N=100;
11
12 % Resolution par differences finies
13 h=1/(N+1);
14 un=ones(N-1,1);
15 A=2*eye(N)-diag(un,-1)-diag(un,1);
16 B=diag(c(h*[1:N]));
17 M=1/h^2*A+B;
18 rhs=[f(h*[1:N]')];rhs(N)=rhs(N)-1/h^2;
19 U=M\rhs;
20
21 % Trace
22 x=linspace(0,1,N+2);
23 plot(x,[0;U;-1])
24 hold on
25 plot(x,uex(x),'r--')
26 legend('Solution approchee','Solution exacte')
27 xlabel('x')
28 set(gca,'FontSize',14)
29 set(gca,'LineWidth',2)
30
31 % Calcul de l'integrale
32 V=[0;U;-1];
33 I=sum(V(1:end-1))*h
```

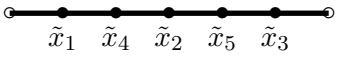
La valeur numérique demandée vaut $\boxed{-0.19765}$.

À travailler pendant la séance de TD

Exercice 2. On considère le problème aux limites étudié en cours

$$\begin{cases} -u''(x) = f(x), \ x \in]0,1[, \\ u(0) = \alpha, \quad u(1) = \beta, \end{cases}$$

On se place dans le cas d'une discrétisation de pas $h = \frac{1}{6}$, et on numérote les nœuds de la manière suivante



L'approximation par différences finies de u au nœud $\tilde{x}_i$ est notée $\tilde{u}_i$. Enfin, le vecteur $\tilde{U}$ désigne $(\tilde{u}_1, \tilde{u}_2, \ldots, \tilde{u}_5)^\intercal$, $\tilde{u}_i$ désignant l'approximation obtenue pour $u(\tilde{x}_i)$.
Écrire le système linéaire $\tilde{A}\tilde{U} = \tilde{F}$ obtenu avec cette numérotation.

Correction.
À l'aide de la discrétisation de la dérivée seconde à 3 points, on obtient le système linéaire

$\tilde{A}\tilde{U} = \tilde{F}$ avec

$$\tilde{A} = \frac{1}{h^2}\begin{pmatrix} 2 & 0 & 0 & -1 & 0 \\ 0 & 2 & 0 & -1 & -1 \\ 0 & 0 & 2 & 0 & -1 \\ -1 & -1 & 0 & 2 & 0 \\ 0 & -1 & -1 & 0 & 2 \end{pmatrix} \quad \textit{et} \quad \tilde{F} = \begin{pmatrix} f(\tilde{x}_1) + \frac{\alpha}{h^2} \\ f(\tilde{x}_2) \\ f(\tilde{x}_3) + \frac{\beta}{h^2} \\ f(\tilde{x}_4) \\ f(\tilde{x}_5) \end{pmatrix}.$$

On voit que la numérotation influe sur la structure de la matrice, même si les problèmes sont équivalents.

N.B. (non demandé dans l'exercice) Si l'on introduit la matrice de permutation P :

$$P = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix},$$

alors $P\tilde{U} = U$ avec $U = (\tilde{u}_1, \tilde{u}_4, \tilde{u}_2, \tilde{u}_5, \tilde{u}_3)^\intercal$, qui représente le vecteur inconnu avec la numérotation habituelle de gauche à droite. Ainsi, on peut écrire

$$P\tilde{A}P^{-1}U = P\tilde{F} = (f(h), f(2h), f(3h), f(4h), f(5h))^\intercal.$$

La matrice $A = P\tilde{A}P^{-1}$ est la matrice habituelle

$$A = \frac{1}{h^2}\begin{pmatrix} 2 & -1 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & -1 & 2 \end{pmatrix}.$$

Ainsi $\tilde{A}$ et A sont semblables, dont ont mêmes valeurs propres : en particulier $\tilde{A}$ est définie-positive.

Exercice 3. Soit $f \in \mathscr{C}^2([0,1], \mathbb{R})$ et $c > 0$. Pour $N \in \mathbb{N}$, on pose

$$h = \frac{1}{N+1}, \quad x_i = ih \quad (0 \leqslant i \leqslant N+1).$$

On considère le problème discret $MU = F$, avec

$$M = \frac{1}{h^2}\begin{pmatrix} 1 & -1 & & & \\ -1 & 2+ch^2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2+ch^2 & -1 \\ & & & -1 & 1 \end{pmatrix}, \quad U = \begin{pmatrix} u_0 \\ u_1 \\ \vdots \\ u_{N+1} \end{pmatrix}, \quad F = \begin{pmatrix} 0 \\ f(x_1) \\ \vdots \\ f(x_N) \\ 0 \end{pmatrix}.$$

1. On suppose que u_i approche $u(x_i)$, où u est une fonction de classe $\mathscr{C}^2([0,1], \mathbb{R})$. Écrire le problème aux limites satisfait par u, dont le modèle précédent est une discrétisation.

2. Montrer que, pour $V = (v_0, v_1, \ldots, v_{N+1})^\intercal \in \mathbb{R}^{N+2}$,

$$(MV, V) = c \sum_{i=1}^{N} v_i^2 + \frac{1}{h^2} \sum_{i=0}^{N} (v_{i+1} - v_i)^2 .$$

En déduire que M est définie positive.

Correction.

1. *La ligne 1 du système linéaire équivaut à*

$$\frac{u_1 - u_0}{h} = 0,$$

qui est une discrétisation de la condition $u'(0) = 0$. De même, la dernière ligne correspond à $u'(1) = 0$. Quant aux lignes intermédiaires, elles s'écrivent :

$$\frac{-u_{i-1} + 2u_i - u_{i+1}}{h^2} + cu_i = f(x_i),$$

qui discrétise l'équation suivante sur $[0, 1]$.

$$-u''(x) + cu(x) = f(x),$$

Le problème aux limites s'écrit dont

$$\begin{cases} -u''(x) + cu(x) = f(x), \ x \in]0, 1[, \\ u'(0) = 0, \quad u'(1) = 0. \end{cases}$$

2. *Un calcul explicite du produit scalaire conduit directement à*

$$(MV, V) = c \sum_{i=1}^{N} v_i^2 + \frac{1}{h^2} \sum_{i=0}^{N} (v_{i+1} - v_i)^2 .$$

Cette expression montre clairement que M est semi-définie positive. Par ailleurs, si $(MV, V) = 0$, alors $v_i = 0$ pour $i = 1, \ldots, N$. Mais on a aussi $v_1 - v_0 = 0$ et $v_{N+1} - v_N = 0$ qui fournit $v_0 = v_{N+1} = 0$, et par suite $V = 0$.

Simulation avec `Matlab`

Exercice 4. On considère le problème aux limites

$$\begin{cases} -u''(x) + \alpha u(x)^3 = f(x), \ x \in]0, 1[, \\ u(0) = 0, \quad u(1) = 0, \end{cases}$$

où α est une constante strictement positive, et $f \in \mathscr{C}([0, 1])$ une fonction donnée.

L'approximation par différences finies de pas $h = 1/(N + 1)$ conduit à l'écriture vectorielle

$$AU + \alpha[U]^3 = F,$$

où A est la matrice tridiagonale du laplacien avec conditions de Dirichlet en dimension 1, F le vecteur de composantes $f(ih)$ pour $i = 1, 2, \ldots, N$; la notation $[U]^3$ désigne le vecteur dont les composantes sont U_i^3.

On propose la méthode suivante pour la résolution de ce problème :

$$\left| \begin{array}{l} U^{(0)} = 0 \in \mathbb{R}^N, \\ \text{Pour } n \geqslant 0, \quad U^{(n+1)} = A^{-1}\left(F - \alpha[U^{(n)}]^3\right). \end{array} \right. \tag{3}$$

On considérera le cas test suivant :

$$f(x) = \pi^2 x \sin(\pi x) - 2\pi \cos(\pi x) + \alpha \left(x \sin(\pi x)\right)^3$$

correspondant à la solution exacte $u(x) = x \sin(\pi x)$.

Programmer cette méthode pour $N = 100$, et les deux valeurs $\alpha = 10$ et $\alpha = 23$. Comment interpréter le résultat ?

Correction. *Le programme suivant met en œuvre la méthode proposée pour la résolution du système non-linéaire $AU + \alpha[U]^3 = F$. Dans le cas $\alpha = 10$, la méthode converge vers la solution exacte, mais dans le cas $\alpha = 23$, il y a divergence (explosion de la solution numérique). Cela s'interprète de la façon suivante : selon la valeur de α, l'itération de point fixe (3) converge ou non (la fonction d'itération n'est pas toujours contractante). Il serait possible de remédier à ce problème en proposant une autre méthode itérative, par exemple la méthode de Newton.*

```
clear
close all

% Cas-test
alpha=10;
f=inline('pi^2*x.*sin(pi*x)-2*pi*cos(pi*x)+alpha*(x.*sin(pi*x)
    ).^3' ...
   ,'x','alpha');
uex=inline('x.*sin(pi*x)');

% Parametres
N=100;
x=linspace(0,1,N+2);

% Resolution par differences finies
h=1/(N+1);
un=ones(N-1,1);
A=2*eye(N)-diag(un,-1)-diag(un,1);
M=1/h^2*A;
rhs=[f(h*[1:N]',alpha)];
% ... iterations
    U=zeros(N,1);
    for i=1:100
        U=M\(rhs-alpha*U.^3);
        clf
        plot(x,[0;U;0])
        hold on
        plot(x,uex(x),'r--')
        legend('Solution approchee','Solution exacte')
```

```
29          title(['Iteration ',num2str(i)])
30          xlabel('x')
31          axis([0,1,0,1])
32          set(gca,'FontSize',14)
33          set(gca,'LineWidth',2)
34          erreur=norm([0;U;0]-uex(x'),'inf')
35          pause(.1)
36      end
```

ÉCOLE CENTRALELYON

Année 2018-2019

Analyse numérique – TD 6

Résolution numérique de l'équation de transport

À préparer avant la séance de TD

Exercice 1. On considère le problème de transport suivant :

$$\begin{cases} \partial_t u(x,t) + c\partial_x u(x,t) = 0, & (x,t) \in \mathbb{R} \times [0,T], \\ u(x,0) = e^{-x^2}, & x \in \mathbb{R}. \end{cases} \tag{$\mathcal{T}_1$}$$

avec $c \in [0,1]$. Afin d'approcher la solution sur $[-5,5]\times[0,T]$ par la méthode des différences finies, on introduit la discrétisation en espace : $x_i = -5 + (i-1)\Delta x$ pour $i \in \{1,\dots,N_x\}$ avec $N_x \geq 2$ et $\Delta x = 10/(N_x - 1)$. On considère le schéma suivant :

$$\begin{cases} \dfrac{U_i^{n+1} - U_i^n}{\Delta t} + c\dfrac{U_i^{n+1} - U_{i-1}^{n+1}}{\Delta x} = 0, & \forall i \in \{2,\dots,N_x\}, \forall n \in \mathbb{N}, \\ U_1^n = 0, & \forall n \in \mathbb{N}. \end{cases} \tag{IG}$$

1. En posant $\beta = \frac{c\Delta t}{\Delta x}$ donner la matrice $\mathcal{A}$ telle que le schéma s'écrive sous forme matricielle $\mathcal{A}\mathbf{U}^{(n+1)} = \mathbf{U}^{(n)}$.
2. Écrire une fonction matlab `A = matriceA(Nx,beta)` qui prend en entrée le nombre de points de discrétisation en espace et β, et qui renvoie la matrice $\mathcal{A}$.
3. Mettre en œuvre le schéma numérique avec $N_x = 1000$ et $T = 4$. Tester différentes valeurs de c.
4. En prenant $c = \frac{1}{2}$ et $\Delta t = 0,01$, vérifier que l'approximation numérique de l'intégrale suivante, obtenue par la méthode des rectangles à gauche

$$I = \int_{-5}^{5} u(x,4)\,\mathrm{d}x.$$

vaut 1.7724.

Correction.

1. *Pour $i > 1$ on a $(1+\beta)U_i^{n+1} - \beta U_{i-1}^{n+1} = U_i^n$ et pour $i = 1$, on peut écrire $(1+\beta)U_1^{n+1} = U_1^n$. Si $\mathbf{U}^n = (U_1^{(n)},\dots,U_{N_x}^{(n)})^\intercal$, la matrice $\mathcal{A}$ intervenant s'écrit*

$$\mathcal{A} = \begin{pmatrix} 1+\beta & & & \\ -\beta & 1+\beta & & \\ & \ddots & \ddots & \\ & & -\beta & 1+\beta \end{pmatrix}.$$

N.B. le coefficient $\mathcal{A}_{11}$ peut être en fait remplacé par tout nombre non nul.

2.
```
function A = matriceA(Nx,beta)
A = (1+beta)*eye(Nx) - beta*diag(ones(Nx-1,1),-1);
```

3.

```
clear
close all

% Parametres
Nx = 1000;
Nt = 400;
T = 4;
c = 0.5;

% Pas
dx = 10/(Nx-1);
dt = T/Nt;
beta = c*dt/dx;



A = matriceA(Nx,beta);
x = linspace(-5,5,Nx)';

% Algorithme
U=exp(-x.^2);
U(1) = 0;
M = zeros(Nx,Nt);
M(:,1) = U;
for n = 1 : Nt
    U = A\U;
    M(:,n) = U;
    plot(x,U)
    axis([-5 5 0 1.1])
    pause(0.01);
end

% Integrale par les rectangle a gauche
I = sum(U(1:end-1))*dx;
I
```

4. *La valeur demandée est $I = 1.7724$ par la méthode des rectangles à gauche.*

À travailler pendant la séance de TD

Exercice 2. On étudie le Schéma de Lax-Freidrichs appliqué au problème $\mathcal{T}_1$:

$$\frac{2U_i^{n+1} - U_{i+1}^n - U_{i-1}^n}{2\Delta t} + c\frac{U_{i+1}^n - U_{i-1}^n}{2\Delta x} = 0, \quad \forall i \in \{1, \dots, N_x\}, \forall n \in \mathbb{N}, \tag{LF}$$

en utilisant la convention $U_0^n = U_{N_x+1}^n = 0$ pour tout n.

1. Écrire le schéma (LF) sous forme matricielle $\mathbf{U}^{n+1} = A\mathbf{U}^n$ en précisant la matrice A en fonction de $\beta = \frac{c\Delta t}{\Delta x}$.
2. Calculer $|||A|||_\infty$. En déduire une condition suffisante sur β pour que le schéma (LF) soit stable en norme ∞.

3. Soit u la solution exacte de $\mathcal{T}_1$. Calculer les développement limités de $u(x, t+\Delta t)$, $u(x+\Delta x, t)$ et $u(x-\Delta x, t)$ au voisinage de (x,t) à l'ordre 3. Sous l'hypothèse $\Delta x/\Delta t \leq \alpha$ avec $\alpha > 0$, montrer que le schéma est consistant et donner ses ordres de consistance.

*4. Montrer que si $\Delta x/\Delta t = c$, alors le schéma est d'ordre 2 en espace et en temps.

Correction.

1. *On a*
$$2U_i^{n+1} = (1-\beta)U_{i+1}^n + (1+\beta)U_{i-1}^n.$$
En posant à nouveau $\mathbf{U}^n = (U_1^{(n)}, \dots, U_{N_x}^{(n)})^\intercal$, *on obtient matriciellement,*
$$\mathcal{A} = \frac{1}{2}\begin{pmatrix} 0 & 1-\beta & & & \\ 1+\beta & 0 & 1-\beta & & \\ & \ddots & \ddots & \ddots & \\ & & & & 1-\beta \\ & & & 1+\beta & 0 \end{pmatrix}.$$

2. *D'après la formule* $|||A|||_\infty = \max_i \sum_j A_{ij} = \frac{|1+\beta|+|1-\beta|}{2}$. *Si* $\beta \in [-1,1]$, *alors* $|||A|||_\infty = \frac{1+\beta+1-\beta}{2} = 1$. *D'après le cours, cela implique la stabilité en norme* ∞.

3. *Voici les DL demandés :*
$$u(x,t+\Delta t) = u(x,t) + \Delta t\partial_t u(x,t) + \frac{\Delta t^2}{2}\partial_{tt}u(x,t) + O(\Delta t^3)$$
$$u(x+\Delta x,t) = u(x,t) + \Delta x\partial_x u(x,t) + \frac{\Delta x^2}{2}\partial_{xx}u(x,t) + O(\Delta x^3)$$
$$u(x-\Delta x,t) = u(x,t) - \Delta x\partial_x u(x,t) + \frac{\Delta x^2}{2}\partial_{xx}u(x,t) + O(\Delta x^3)$$
L'erreur de consistance est
$$\varepsilon_i^n = \frac{2u(x_i,t_{n+1}) - u(x_{i+1},t_n) - u(x_{i-1},t_n)}{2\Delta t} + c\frac{u(x_{i+1}t_n) - u(x_{i-1},t_n)}{2\Delta x}$$
à l'aide des DL précédent,
$$\begin{aligned}\varepsilon_i^n &= \partial_t u(x_i,t_n) + \Delta t\partial_{tt}u(x_i,t_n) - \frac{\Delta x^2}{\Delta t}\partial_{xx}u(x_i,t_n) + O(\Delta t^2) + O(\frac{\Delta x^3}{\Delta t}) \\ &\quad + c\partial_x u(x_i,t_n) + O(\Delta x^2).\end{aligned}$$
$$\varepsilon_i^n = \Delta t\partial_{tt}u(x_i,t_n) - \frac{\Delta x^2}{\Delta t}\partial_{xx}u(x_i,t_n) + O(\Delta x^2) + O(\Delta t^2) + O(\frac{\Delta x^3}{\Delta t}).$$
En supposant $\Delta x/\Delta t \leq \alpha$, *on a* $\frac{\Delta x^2}{\Delta t} \leq \alpha\Delta x$ *et* $O(\frac{\Delta x^3}{\Delta t}) = O(\Delta x^2)$. *Ainsi*
$$\varepsilon_i^n = O(\Delta t) + O(\Delta x) = O(\Delta t + \Delta x).$$
Le schéma est consistant d'ordre 1 en temps et 1 en espace.

4. *Si on suppose $\Delta x/\Delta t = c$ on a*

$$\varepsilon_i^n = \Delta t \left(\partial_{tt} u(x_i, t_n) - c^2 \partial_{xx} u(x_i, t_n)\right) + O(\Delta x^2) + O(\Delta t^2).$$

En dérivant l'EDP par rapport à t et par rapport à x on obtient

$$\partial_{tt} u + c\partial_{xt} u = 0,$$
$$\partial_{tx} u + c\partial_{xx} u = 0,$$

On en déduit que

$$\partial_{tt} u - c^2 \partial_{xx} u = 0,$$

et donc

$$\varepsilon_i^n = O(\Delta x^2 + \Delta t^2).$$

L'ordre est 2 en temps et en espace.

Exercice 3. On considère le problème de transport

$$\begin{cases} \partial_t u(x,t) + 2xt\partial_x u(x,t) = 0, \quad (x,t) \in \mathbb{R} \times [0,T], \\ u(x,0) = x^2(1-x)^2 \chi_{[0,1]}(x), \quad x \in \mathbb{R}. \end{cases} \tag{$\mathcal{T}_2$}$$

1. Donner l'équation des caractéristiques X associées à ce problème et la résoudre.
2. Quelle est l'équation différentielle vérifiée par $\varphi(t) = u(X(t), t)$?
3. En déduire la solution explicite de $(\mathcal{T}_2)$.

Correction.

1. *Le champ de vitesse est $v(x,t) = 2xt$. Ainsi $X'(t) = v(X(t),t) = 2tX(t)$. les solutions de cette EDO linéaire sont*

$$X(t) = ke^{t^2}, \quad k \in \mathbb{R}.$$

2. *$\varphi(t) = u(ke^{t^2}, t)$. On dérive cette fonction :*

$$\varphi'(t) = \partial_x u(ke^{t^2}, t) 2ke^{t^2} t + \partial_t u(ke^{t^2}, t) = \partial_x u(x,t) 2xt + \partial_t u(x,t) = 0$$

en posant $x = ke^{t^2}$.

3. *Pour toute caractéristique X, $t \mapsto u(X(t),t)$ est constante.*
Soit $(x_0, t_0) \in \mathbb{R} \times [0,T]$ la caractéristique qui passe par x_0 en t_0 est $X(t) = x_0 e^{t^2 - t_0^2}$.
On a

$$u(x_0, t_0) = u(X(t_0), t_0) = u(X(0), 0) = u_0(X(0)) = u_0(x_0 e^{-t_0^2}).$$

Ainsi

$$u(x,t) = u_0(xe^{-t^2}) = xe^{-t^2}\left(1 - xe^{-t^2}\right)\chi_{[0,1]}\left(xe^{-t^2}\right), \quad \forall x \in \mathbb{R}, \forall t \in [0,T]$$

Exercice 4. On va utiliser le schéma explicite décentré à gauche

$$\begin{cases} \dfrac{U_i^{n+1}-U_i^n}{\Delta t}+2x_it_n\dfrac{U_i^n-U_{i-1}^n}{\Delta x}=0, & \forall i\in\{2,\dots,N_x\},\forall n\in\mathbb{N}, \\ U_1^n=0, & \forall n\in\mathbb{N}. \end{cases} \tag{EG}$$

pour résoudre le problème $(\mathcal{T}_2)$ sur le domaine $[0,10]$ en prenant $T=\frac{3}{2}$.

1. Donner une écriture matricielle du schéma de la forme $\mathbf{U}^{n+1}=\mathcal{A}_n\mathbf{U}^n$. On fera apparaitre $\beta=\Delta t/\Delta x$.
2. Mettre en œuvre ce schéma. On programmera une visualisation dynamique de `U` ainsi qu'un affichage direct de la solution à l'aide des commandes `surf`, `imagesc` ou `contour`.
3. Quelle est la valeur maximale de β telle que l'on ait convergence du schéma ?

*4. En utilisant la solution exacte déterminée dans l'exercice 2, mesurer les ordres de convergence de la méthode en évaluant l'erreur en norme infinie au temps final $T=\frac{3}{2}$ pour différentes valeur de N_x et N_t respectant la condition CFL de la question précédente.

Correction.

1. On a

$$\mathcal{A}_n=\mathcal{I}-2\beta t_n\begin{pmatrix} x_1 & & & \\ -x_2 & x_2 & & \\ & \ddots & \ddots & \\ & & -x_n & x_n \end{pmatrix}.$$

2.

```
function U = SchemaEG(Nx,Nt,T,U0)


dt = T/Nt;
dx = 10/(Nx-1);
beta = dt/dx;
disp(['beta = ' num2str(beta)])
x = linspace(0,10,Nx)';

% Matrice d'iteration
I = sparse(eye(Nx));
D = diag(x) - diag(x(2:Nx),-1);
D = sparse(D);

% Initialisation
U = zeros(Nx,Nt+1);


U(:,1) = U0;

% Algorithme
```

```
for n = 1 : Nt
    tn = (n-1)*dt;
    A = I - 2*tn*beta*D;
    U(:,n+1) = A*U(:,n);
end
```

Script pour l'affichage de la solution :

```
clear
close all

% Parametres
Nx = 50;
Nt = 300;
T = 3/2;

% Axes
x = linspace(0,10,Nx)';
t = linspace(0,T,Nt+1)';

% Solution
U0 = x.^2.*(1-x).^2.*(x>=0).*(x<=1);
U = SchemaEG(Nx,Nt,T,U0);

% Affichages statiques
figure(1)
imagesc(x,t,U')
axis xy
title('Solution u(x,t)')
xlabel('x')
ylabel('t')
colorbar

figure(2)
contour(x,t,U',20)
title('Solution u(x,t) : courbe de niveaux')
xlabel('x')
ylabel('t')
drawnow

figure(3)
surf(x,t,U','EdgeColor','none')
title('Solution u(x,t) : graphe 3D')
xlabel('x')
ylabel('t')
zlabel('u(x,t)')
drawnow

% Affichage dynamique
figure(4)
```

```
for n = 1 : 10 : size(U,2)
    plot(x,U(:,n))
    xlabel('x')
    ylabel('u(x,t)')
    title(['n = ' num2str(n-1)])
    axis([0 10 0 0.2])
    pause(0.001)
end
```

3. *On trouve à la main un β max d'environ $0,055$. En réalité, la CFL ici demande que $v(x,t)\beta \leq 1$ soit $2 \times 10 \times 3/2 \times \beta \leq 1$ soit $\beta \leq 1/30$ i.e. $\beta \leq 0,03333$*

4. *Voir un algorithme de test de convergence vers la solution en $T = 3/2$:*

```
clear
close all

% Parametres
Nx = 500;
Nt = 5*Nx;
T = 3/2;

u0_fun = @(x) x.*(1-x).*(x>=0).*(x<=1);
u0_fun = @(x) 1*(x>0).*(x<1).*exp(1./((2*x-1).^2-1));



% Axes
x = linspace(0,10,Nx)';
t = linspace(0,T,Nt+1)';

% Solution approchee
U0 = u0_fun(x);
U0(isnan(U0)) = 0;
U = SchemaEG(Nx,Nt,T,U0);
u = U(:,Nt+1);

% Solution eplicite
uex = u0_fun(x*exp(-T^2));

% Comparaison graphique
plot(x,u,x,uex)

Nxtest = [2000 5000 10000 20000];

E = [];
for Nx = Nxtest
    Nx
    Nt = 5*Nx;
    x = linspace(0,10,Nx)';
    U0 = u0_fun(x);
```

```
38      U0(isnan(U0)) = 0;
39      U = SchemaEG(Nx,Nt,T,U0);
40      u = U(:,Nt+1);
41      uex = u0_fun(x*exp(-T^2));
42
43      x = linspace(0,10,Nx)';
44      E = [E max(abs(u-uex))];
45  end
46
47  figure
48  plot(log(Nxtest),log(E),'-o')
49  polyfit(log(Nxtest),log(E),1)
```

Analyse numérique – TD 7

TD de synthèse : résolution d'une EDP non linéaire en dimension 2

À préparer avant la séance de TD

On considère le problème aux limites suivant, posé dans $\omega =]0,1[\times]0,1[$:

$$\begin{cases} -\Delta u(x,y) = f(x,y), \text{ dans } \Omega, \\ u(x,y) = 0, \text{ sur } \partial\Omega, \end{cases}$$

où la fonction f est donnée par

$$f(x,y) = 2\pi^2 \sin(\pi x)\sin(\pi y).$$

Résoudre ce problème à l'aide de la méthode de différences finies utilisant le laplacien à 5 points.

On vérifiera la solution calculée en remarquant que la solution exacte correspondante est donnée par

$$u(x,y) = \sin(\pi x)\sin(\pi y).$$

Correction. *La matrice est construite par blocs, comme indiqué dans le cours 5.*

```
function A=Lap2D(N)
h=1/(N+1);
% Construction de la matrice
H=4*eye(N)-diag(ones(N-1,1),1)-diag(ones(N-1,1),-1);
I=eye(N);
A=zeros(N*N,N*N);
for i = 1 : N
    A(1+(i-1)*N:i*N,1+(i-1)*N:i*N) = H;
    if i>1
        A(1+(i-1)*N:i*N,1+(i-2)*N:(i-1)*N) = -I;
    end
     if i<N
        A(1+(i-1)*N:i*N,1+(i)*N:(i+1)*N) = -I;
    end
end

A=A/h^2;
```

Le programme suivant résout le système linéaire et compare la solution obtenue à la solution exacte (l'erreur est de l'ordre de 3%).

```
close all
clear
N=50;
A=Lap2D(N);
```

```
x=linspace(0,1,N+2);
y=linspace(0,1,N+2);

% Construction du second membre et de la solution exacte
F=zeros(N*N,1);
Uex=zeros(N*N,1);
k=0;
for j=1:N
    for i=1:N
        k=k+1;
        F(k)=2*pi^2*sin(pi*x(i+1)).*sin(pi*y(j+1));
        Uex(k)=sin(pi*x(i+1)).*sin(pi*y(j+1));
    end
end

% Resolution du systeme
U=A\F;

% Comparaison avec la solution exacte
Erreur=norm(U-Uex,'inf')/norm(U,'inf');
disp(['Erreur relative en norme infinie : ',num2str(Erreur)]);

% Reomposition : vecteur N^2 --> matrice N x N
Umatrice=reshape(U,N,N);

% Ajout des zeros au bord
colzero=zeros(N,1);
ligzero=zeros(1,N+2);
Ucomplet=[ligzero;[colzero,Umatrice,colzero];ligzero];

% Post-traitement graphique
surf(x,y,Ucomplet)
xlabel('x')
ylabel('y')
```

À travailler pendant la séance de TD

Exercice 1. On considère le problème aux limites suivant, posé dans $\Omega =]0,1[\times]0,1[$:

$$\begin{cases} -\Delta u(x,y) + 8u(x,y)^3 = f(x,y), \text{ dans } \Omega, \\ u(x,y) = g(x,y), \text{ sur } \partial\Omega. \end{cases}$$

On choisira les données f et g comme ceci (la constante k sera précisée plus loin) :

$$f(x,y) = (1+k^2)\pi^2 \sin(\pi y)\sin(k\pi x) + 8\Big(\sin(\pi y)\sin(k\pi x) + \sin(\pi y)\sin(\pi\tfrac{x}{2})\cos(\pi k)\Big)^3 + \frac{5}{4}\pi^2 \cos(k\pi)\sin(\pi y)\sin(\pi\tfrac{x}{2}).$$

$$g(x,y) = \begin{cases} \sin(\pi y)\big(\sin(k\pi)+\cos(k\pi)\big), & \text{si } x = 1, \\ 0 & \text{sinon.} \end{cases}$$

Défi 1. Pour $k = 0.7$, calculer une approximation de la valeur $u\left(\frac{1}{2},\frac{1}{2}\right)$.

Défi 2. Pour $k = 0.7$, calculer une approximation de l'intégrale

$$\int_\Omega u(x,y)\,\mathrm{d}x\,\mathrm{d}y.$$

Défi 3. Trouver une approximation de la valeur de $k \in [-1,1]$ pour laquelle l'intégrale

$$\int_\Omega u(x,y)\,\mathrm{d}x\,\mathrm{d}y$$

est minimale.

Correction.

N.B. *Les données ont été choisies telles que la solution exacte du problème s'écrive*

$$u(x,y) = \sin(k\pi x)\sin(\pi y) + \sin\left(\pi\tfrac{x}{2}\right)\sin(\pi y)\cos(\pi k).$$

Ainsi, on a accès aux valeurs exactes correspondant aux défis proposés :

- ◇ *Valeur au centre :*

$$u\left(\tfrac{1}{2},\tfrac{1}{2}\right) = \sin\left(k\tfrac{\pi}{2}\right) + \tfrac{\sqrt{2}}{2}\cos(k\pi) \simeq 0.475379586410914.$$

- ◇ *Intégrale :*

$$\int_\Omega u(x,y)\,\mathrm{d}x\,\mathrm{d}y = \frac{4k\cos(k\pi) - 2\cos(k\pi) + 2}{k\pi^2} \simeq 0.221426127561213.$$

- ◇ *Valeur optimale de k : il s'agit de la racine dans $[-1,1]$ de la fonction dérivée de l'intégrale, qui résout l'équation*

$$\cos(k\pi) + k\pi\sin(\pi x) - 2\pi x^2\sin(\pi x) = 1.$$

Cette équation ne peut pas être résolue à la main ; à l'aide de la méthode de Newton (ou de la commande `fminsearch` *de* `matlab`*), on obtient*

$$k_{\min} \simeq -0.923268196451616.$$

Les fonctions suivantes définissent les données (on a ajouté la variable k comme argument en vue du défi 3).

```
1  function z=f(x,y,k)
2
3  z=(1+k^2)*pi^2*sin(pi*y)*sin(k*pi*x)+8*(sin(pi*y)*sin(k*pi*x)
       ...
4    +sin(pi*y)*sin(pi*x/2)*cos(pi*k))^3 ...
5    +5/4*pi^2*cos(pi*k)*sin(pi*y)*sin(pi*x/2);
```

```
function z=g(y,k)

z=sin(pi*y)*sin(pi*k)+sin(pi*y)*cos(pi*k);
```

Défi 1. *Pour la résolution de l'EDP non-linéaire, une itération directe comme proposée au TD5 :*

$$AU^{(n+1)} = F - 8[U^{(k)}]^3,$$

converge très lentement (on rappelle que $[V]^3$ désigne le vecteur de coordonnées V_i^3). On doit lui préférer la méthode de Newton, qui s'écrit ici

$$U^{(n+1)} = U^{(n)} - \left(A + 24\,\mathrm{diag}\left([U^{(n)}]^2\right)\right)^{-1}\left(AU^{(n)} + 8[U^{(n)}]^3 - F\right).$$

```
close all
clear
% Parametres
N=51;
h=1/(N+1);
k=0.7;
%... pour Newton
tol=1e-5;
itermax=10;

% Matrice des differences finies
A=Lap2D(N);

% Construction du second membre
F=zeros(N*N,1);
q=0;
for i=1:N
    for j=1:N
        q=q+1;
        F(q)=f(i*h,j*h,k);
        if (i==N)
            F(q)=F(q)+g(j*h,k)/h^2;
        end
    end
end

%% ======== DEFI 1 ======= %%

% Resolution par iteration directe
Err=1+tol;
iter=0;
V=A\F;
while (Err>tol&iter<itermax)
    iter=iter+1;
    Vnew=A\(F-8*V.^3);
    Err=norm(V-Vnew,'inf');
```

```
    V=Vnew;
end
disp('=== Defi1 ===')
disp(['Iteration directe : ',num2str(V((N^2+1)/2),10)])

% Resolution par la methode de Newton
Err=1+tol;
iter=0;
U=A\F;
while (Err>tol&iter<itermax)
    iter=iter+1;
    J=A+24*diag(U.^2);
    Unew=U-J\(A*U+8*U.^3-F);
    Err=norm(U-Unew,'inf');
    U=Unew;
end

disp(['Newton : ',num2str(U((N^2+1)/2),10)])
```

N.B. *il est préférable ici de choisir N impair pour que le point $\left(\frac{1}{2},\frac{1}{2}\right)$ soit un nœud de la grille.*

Pour $N = 51$, on trouve les valeurs suivantes :

- ◇ *Par itération directe :* 0.4754706631.
- ◇ *Par la méthode de Newton :* 0.4754694536.

Défi 2. *Pour le calcul de l'intégrale, on doit préférer la méthode des trapèzes (d'ordre 2) à la méthode des rectangles (d'ordre 1). En revanche, il n'est pas très pertinent d'utiliser une méthode d'ordre plus élevé, car la discrétisation de l'EDP est elle-même d'ordre 2.*

```
%% ======== DEFI 2 ======= %%

disp('=== Defi2 ===')
Irect=sum(U)*h^2;
Itrap=sum(U)*h.^2+h^2/2*sum(g(h*[1:N],k));
disp(['Rectangles : ',num2str(Irect,10)])
disp(['Trapezes : ',num2str(Itrap,10)])
```

Pour $N = 51$, on trouve les valeurs suivantes :

- ◇ *Par la méthode des rectangles :* 0.2199932309.
- ◇ *Par la méthode des trapèzes :* 0.2213470545.

Défi 3. *Pour les valeurs de $k < -0.6$, la méthode d'itération directe ne converge plus (l'itération n'est plus contractante), il faut donc nécessairement utiliser la méthode de Newton ! On commence par mettre dans une fonction la résolution de l'EDP et le calcul de l'intégrale :*

```
function Integrale=resol(k,N,tol,itermax)

h=1/(N+1);
```

```
% Matrice des differences finies
A=Lap2D(N);

% Construction du second membre
F=zeros(N*N,1);
for i=1:N
    for j=1:N
        F(j+(i-1)*N)=f(i*h,j*h,k);
        if (i==N)
            F(j+(i-1)*N)=F(j+(i-1)*N)+g(j*h,k)/h^2;
        end
    end
end

% Resolution par la methode de Newton

Err=1+tol;
iter=0;
U=A\F;
while (Err>tol&iter<itermax)
    iter=iter+1;
    J=A+24*diag(U.^2);
    Unew=U-J\(A*U+8*U.^3-F);
    Err=norm(U-Unew,'inf');
    U=Unew;
end

% Calcul de l'integrale
Integrale=sum(U)*h.^2+h^2/2*sum(g(h*[1:N],k));
```

Si l'on fait un graphe de la fonction $k \mapsto \int_\Omega u(x,y)\,\mathrm{d}x\,\mathrm{d}y$, on s'aperçoit qu'elle est unimodale sur $[-1,0]$:

```
% Parametres
N=10;
Nb=100;
tol=1e-15;
itermax=10;

% Initialisatoin
liste_k=linspace(-1,1,Nb);
liste_Vals=zeros(Nb,1);

% Boucle
for i=1:Nb
    liste_Vals(i)=resol(liste_k(i),N,tol,itermax);
end

% Graphe
```

```
plot(liste_k,liste_Vals);
```

On peut mettre en œuvre la méthode du nombre d'or pour trouver le point de minimum (plus facile à mettre en œuvre que la méthode de dichotomie car elle ne réclame pas de calcul de dérivée) :

```
%% ======== DEFI 3 ======= %%

% Parametres
N=50;
tol=1e-5;

% Nombre d'or
gamma=(1+sqrt(5))/2;

% Initialisation
a=-1;
b=0;
c = b - (gamma-1)*(b-a);
d = a+(gamma-1)*(b-a);

% Iteration
iter=0;
while (abs(b-a)>tol)
    iter=iter+1;
    disp(['Iteration ',num2str(iter)]);
    % ... Calculs des valeurs de l'integrale
    Ic=resol(c,N,tol,itermax);
    Id=resol(d,N,tol,itermax);

    % ... Tests et mise a jour
    if (Ic<=Id)
        b=d;
        d=c;
        c=b-(b-a)/gamma;
    else
        a=c;
        c=d;
        d=a+(b-a)/gamma;
    end
 end

disp('=== Defi3 ===')
disp(['Valeur du k optimal : ',num2str((a+b)/2,10)])
```

Pour $N = 50$, on trouve après 24 itérations la valeur suivante :

$$k_{\min} \simeq -0.9232452792.$$

Remarque. *Les calculs sont assez longs, car la résolution du système linéaire devient coûteuse, même pour des valeurs modérées de N (la taille de la matrice est $N^2 \times N^2$). Il est possible d'accélérer les calculs en utilisant le fait que la matrice A est creuse (i.e. a beaucoup de zéros, placés de manière structurée). La commande* `sparse` *de* `matlab` *permet cela. La fonction suivante effectue la même tâche que la fonction* `resol.m`*, mais prend approximativement* 300 *fois moins de temps pour $N = 100$.*

```
function Integrale=resol_sparse(k,N,tol,itermax)

h=1/(N+1);
% Matrice des differences finies
A=delsq(numgrid('S',N+2));
A=A/h^2;

% Construction du second membre
F=zeros(N*N,1);
q=0;
for j=1:N
    for i=1:N
        q=q+1;
        F(q)=f(i*h,j*h,k);
        if (i==N)
            F(q)=F(q)+g(j*h,k)/h^2;
        end
    end
end

% Resolution par la methode de Newton

Err=1+tol;
iter=0;
U=A\F;
while (Err>tol&iter<itermax)
    iter=iter+1;
    J=A+24*sparse(1:N*N,1:N*N,U.^2,N*N,N*N);;
    Unew=U-J\(A*U+8*U.^3-F);
    Err=norm(U-Unew,'inf');
    U=Unew;
end

% Calcul de l'integrale
Integrale=sum(U)*h.^2+h^2/2*sum(g(h*[1:N],k));
```