SCIENCES SUP

# Exercices & Problèmes

Licence • Écoles d'ingénieurs

# ANALYSE NUMÉRIQUE AVEC MATLAB

- ▶ Rappels de cours
- ▶ Méthodes
- ▶ Exercices et problèmes avec corrigés détaillés

*Jean-Louis Merrien*

DUNOD

# ANALYSE NUMÉRIQUE AVEC MATLAB

- Rappels de cours
- Méthodes
- Exercices et problèmes avec corrigés détaillés

***Jean-Louis Merrien***
Maître de conférences à l'INSA de Rennes

DUNOD

Illustration de couverture : *digitalvision*





© Dunod, Paris, 2007
ISBN 978-10-050863-1

# Table des matières

# Introduction

Cet ouvrage s'adresse aux étudiants en licence de mathématique appliquée ou en formation d'ingénieur. Son objectif est de donner au lecteur un outil lui permettant de travailler de manière autonome à l'aide de questions détaillées et progressives, et d'une construction pas à pas des programmes.

Ce choix de faire de la théorie avant de commencer la programmation est indispensable pour appréhender les notions d'analyse numérique mais aussi pour améliorer ses capacités de programmeur ; la programmation demande un peu d'âme... Cette préparation ne dispense pas d'une réflexion sur la manière de programmer une méthode. À cette fin, les exercices en `Matlab` proposent une programmation sous forme de poupées russes. À chaque question, le programme précédent est amélioré et complété. Les résultats intermédiaires sont donnés pour valider cette programmation par morceaux.

En fin de chapitre, les solutions complètes et les programmes sont systématiquement donnés. Naturellement, un livre de cours d'analyse numérique est utile en complément de cet ouvrage et la bibliographie en propose quelques-uns.

Quatres rubriques sont destinées à améliorer et faciliter la recherche des exercices ainsi que leur compréhension :

- une rubrique « Rappel de cours » ;
- une rubrique « Du mal à démarrer ? » donne des pistes pour commencer un exercice ;
- une rubrique « Commentaire » indiquée par  ;
- une rubrique « Ce qu'il faut retenir de cet exercice ».

En fin de volume, deux index permettent d'obtenir rapidement de l'information : un index général et un index des commandes `Matlab`. Pour ce dernier, chaque mot clé a au plus trois références, même si la commande est utilisée beaucoup plus souvent.

Les premiers chapitres peuvent être considérés comme une initiation.

Le premier chapitre rappelle les commandes utiles de `Matlab` pour gérer des tableaux et les commandes élémentaires. Il s'agit donc de savoir utiliser au mieux les tableaux et de s'initier aux premières commandes du graphisme. Ces commandes nécessaires sont insuffisantes pour progresser dans la programmation et il ne faut pas hésiter à consulter fréquemment l'aide en ligne de `Matlab`.

Dans les deuxième et troisième chapitres est abordée la notion centrale de l'analyse numérique : les matrices. En effet, beaucoup de méthodes numériques conduisent à la résolution d'un système linéaire. Les méthodes de résolution de système linéaire ne sont pas détaillées ici. Néanmoins les quelques exercices proposés peuvent servir de base. Plutôt que de multiplier ces exercices dans le

chapitre, certains ont été placés dans d'autres chapitres ; l'index permet de les retrouver. Ensuite, on a choisi de privilégier la notion de conditionnement car les résolutions de systèmes linéaires peuvent conduire à de grosses erreurs numériques. Les exercices proposés permettent de constater que le conditionnement est une notion originale qui n'a rien à voir avec les notions de déterminant ou de valeur propre.

Le chapitre 4 est consacré à l'interpolation ou comment faire passer une courbe par des données mesurées. La première réponse est donnée par l'interpolant de Lagrange. Mais le phénomène de Runge montre qu'en plus de passer par les points, le cahier des charges peut aussi imposer une approximation convenable pour les autres points (lorsqu'on part d'une fonction échantillonnée par exemple). Ce chapitre propose donc une étude des splines cubiques qui répond mieux à cette question.

Dans le chapitre 5, une étude d'erreur est proposée. Il s'agit d'une notion essentielle en analyse numérique où, traditionnellement, la première étape est de montrer l'existence d'une solution unique à un problème sans forcément savoir la calculer, puis, la seconde de construire un problème approché dont la solution est cette fois-ci calculable ; pour finir on majore l'erreur entre les solutions exacte et approchée. Dans certains cas particuliers, les deux solutions peuvent être calculées. L'erreur est connue exactement ; on peut alors mesurer si la majoration est optimale. `Matlab` permet ainsi d'estimer l'ordre d'une méthode.

Dans le chapitre 6, nous répondons à la question : comment approcher des données mesurées par une courbe ? La notion est différente de celle de l'interpolation et on se gardera donc de les confondre. Comme les mesures ne sont pas toujours linéaires, on verra qu'au delà de la régression linéaire, différentes bases peuvent être utilisées.

S'ajoute un chapitre, moins classique dans les cours d'analyse numérique, sur les courbes de Bézier et les polynômes de Bernstein, introduction au dessin et à la conception assistés par ordinateur (DAO, CAO). Le cours y est proposé sous forme d'exercices.

Dans la suite, d'autres outils traditionnels de l'analyse numérique sont abordés : méthodes pour les équations différentielles ou méthodes élémentaires pour les équations aux dérivées partielles. Encore une fois, la présentation est loin d'être exhaustive. Au contraire, elle se propose de mettre en avant quelques problèmes théoriques ou numériques.

Enfin, dans le dernier chapitre, on trouvera des problèmes qui combinent souvent plusieurs des techniques proposées précédemment.

La plupart de ces exercices ont été testés par les étudiants de l'Insa de Rennes. La majorité est même extraite des sujets d'examens qu'on peut réaliser en deux heures avec un peu d'entraînement.

Je remercie mes collègues de l'INSA de Rennes qui ont participé à l'élaboration ou à la correction d'une bonne partie de ces exercices.

# 1

# Petits rappels sur les commandes Matlab

L'objet de ce chapitre est de mettre ou remettre en mémoire quelques commandes Matlab. En particulier, on s'intéresse à la gestion des tableaux et matrices et on rappelle quelques commandes graphiques. Il est fortement conseillé de ne pas négliger ce chapitre qui donne des astuces et des méthodes pour la suite même s'il peut paraître un peu fastidieux ou s'apparentant à de la dactylographie. Par ailleurs, un petit exemple permet de comparer le traitement vectoriel de Matlab à un programmation à l'aide de boucles.

## 1.1 PREMIÈRES COMMANDES

On peut taper plusieurs commandes Matlab sur une même ligne, en les séparant par une virgule. Quelques exemples élémentaires à tester :

➤ Opérations numériques

```
>> 5*6, 2^5
>> 3+5*2^5
```

➤ Comment déclarer des variables (signe =)

```
>> x=2
>> y=x^5
>> y/x
```

➤ Les variables peuvent s'afficher sous différents formats

```
>> a=sqrt(3)
>> format long, b=sqrt(3)
>> a-b
>> format short
>> who
>> clear
>> who
```

Les variables apparaissent aussi dans la fenêtre Workspace. Cette fenêtre permet aussi en cas de problème dans la programmation de connaître les dimensions d'une variable tableau pour voir si elles sont conformes aux prévisions...

➤ Erreurs d'arrondi

Découvrez eps en tapant tapant (>> help eps). Taper :

```
>> 1+eps-1
>> 1+eps/2-1
```

1+eps apparaît comme le plus petit nombre machine strictement supérieur à 1.

## 1.2 MATRICES, VECTEURS, TABLEAUX

En fait Matlab est avant tout un outil matriciel ; il considère un nombre réel comme une matrice 1x1. Tout est donc tableau, et Matlab est particulièrement adapté aux calculs numériques d'algèbre linéaire.

Voici tout d'abord différentes possibilités pour créer ou modifier une matrice :

```
>> a=[1,2,3;4 5 6]
>> a(1,2), a(2,3)
>> a(2,3)=10
>> a'
>> rand(1,3), rand(2)
>> zeros(3), ones(3,2)
>> eye(3), eye(2,3)
>> magic(3)
>> b=[1 4 5], diag(b)
>> whos
```

Découvrez l'aide en ligne soit en utilisant help dans le menu soit en tapant par exemple help magic dans l'interpréteur. Un peu d'anglais est parfois utile...

À noter que l'affichage peut être supprimé en terminant la commande par un point-virgule. C'est utile quand on travaille avec de grosses matrices.

```
>> s1=zeros(20,25) ; s2=zeros(2,3)
```

L'opérateur « : » est très utile pour construire un tableau ou en extraire une partie, une ligne ou une colonne. Testez :

```
>> -3:3
>> x=-3:.3:3
>> x(2:12)
>> x(9:-2:1)
>> x=10:100; x(2), x(10)
>> x(40:5:60)
```

```
>> a=[1:6 ; 2:7 ; 4:9]
>> a, a(1, :), a(:, 2)
>> s=rand(20,5); s(6:7, 2:4)
```

➤ Premier exemple de boucle

```
>> for i=1:2000
for j=1:1000
t(i,j)=i/j ;
end
end
```

Matlab effectue du calcul vectoriel et ce genre d'opération peut être effectué beaucoup plus rapidement. Pour comparer, tapez

```
>> i=(1:2000);
>> j=1./(1:1000);
>> t2=i'*j;
```

C'est bien la même matrice aux erreurs d'arrondis près ; on le vérifie avec (>> max(max((abs(t2-t))))).

➤ Calcul matriciel

Après avoir entré les données, on peut essayer d'effectuer les opérations ci-dessous :

```
>> a=[1 2 3; 4 5 6; 7 8 10] , b=[1 1 1]'
>> 2*a , a/4
>> a+[b,b,b]
>> a*b
>> b*a
>> b'*a
>> b'*b , b*b'
>> a^2
```

mais aussi ces opérations étranges.

```
>> a+1, b+2
>> a.^2, a.*a
>> a.*b
>> 1./a, 1./a.^2
```

➤ Systèmes linéaires

Un système linéaire ax=b peut être résolu en une seule petite commande ; dans l'exemple choisi ici le système admet une solution unique.

```
>> x=a\b
```

Vérifiez en calculant : (>> a*x , a*x-b)
Réessayez avec un autre second membre :

```
>> b=[1 1 0]'
>> x=a\b , a*x , a*x-b
```

On peut aussi résoudre le système linéaire ya=b' par (>>y=b'/a)
Bien sûr l'instruction (>> y=b/a) donne un message d'erreur.

➤ D'autres fonctions matricielles

```
>> det(a), rank(a), inv(a), eig(a)
```

## 1.3 QUELQUES EXEMPLES ÉLÉMENTAIRES DE GRAPHIQUES

➤ Fonctions d'une variable réelle

```
>> x =-10: .001:10 ;
>> plot( x.^2)
>> figure
>> plot( x, x.^2)
>> plot( x, x.*sin(x) )
>> plot(x.*cos(x),x.*sin(x) )
>> comet(x.*cos(x),x.*sin(x) )
>> title('c''est␣joli');
```

➤ Fonctions de deux variables réelles

```
>> [x,y]=meshgrid([0:1:3],[1:0.5:2])
>> [x y]=meshgrid(-3:0.1:3,-4:0.2:5);
>> z = x.^2 + y.^2;
>> mesh(x,y,z)
>> surf(x,y,z)
>> xlabel('x')
>> contour(x,y,z)
```

# 2 Matrices

## RAPPEL DE COURS

Soit $A = (a_{ij})$ une matrice de $\mathbb{R}^{n\times n}$ (ou $\mathbb{C}^{n\times n}$). $A$ est **inversible** s'il existe $B \in \mathbb{R}^{n\times n}$ (ou $\mathbb{C}^{n\times n}$) telle que $AB = BA = I$, notation $B = A^{-1}$.

Le **rang** de $A$ est le nombre de vecteurs colonnes (ou lignes) indépendants ; notation $\text{rg}(A)$. *Le rang est la dimension de la plus grande matrice carrée de déterminant non nul extraite de A.*

Le **noyau** de $A$ est l'ensemble des vecteurs $X$ tels que $AX = 0$ ; notation $\text{Ker}(A)$.

Un réel ou un complexe $\lambda$ est une **valeur propre** de $A$ et $V \in \mathbb{R}^n$ (ou $\mathbb{C}^n$), $V$ non nul, un **vecteur propre associé** si $AV = \lambda V$. *Les valeurs propres de A sont les racines du polynôme* $\det(A - XI) = 0$. L'ensemble des valeurs propres de $A$ est le **spectre** de la matrice ; notation $\sigma(A)$.

*$A \in \mathbb{C}^{n\times n}$ est inversible à l'une des conditions nécessaires et suffisantes suivantes :* $0$ *n'est pas valeur propre ou* $\det(A) \neq 0$ *ou* $Ker(A) = \{0\}$ *ou* $rang(A) = n$.

$A$ est **diagonalisable** dans $\mathbb{C}$ s'il existe une matrice inversible $P \in \mathbb{C}^{n\times n}$ et une matrice diagonale $D$ dans $\mathbb{C}^{n\times n}$ telles que $P^{-1}AP = D$. Dans ce cas les valeurs propres de $A$ sont sur la diagonale de $D$, les vecteurs propres sont les colonnes de $P$. $A$ peut être diagonalisable dans $\mathbb{R}$ si $D$ et $P$ sont dans $\mathbb{R}^{n\times n}$. *$A$ est diagonalisable si et seulement si il existe une base $\{u_1, \ldots, u_n\}$ de vecteurs propres.*

*Décomposition de Jordan : Soit $A \in \mathbb{C}^{n\times n}$, il existe une matrice $P \in \mathbb{C}^{n\times n}$ inversible telle que*

$$P^{-1}AP = \begin{pmatrix} J_{k_1}(\lambda_1) & 0 & \ldots & 0 \\ 0 & J_{k_2}(\lambda_2) & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \ldots & 0 & J_{k_\ell}(\lambda_\ell) \end{pmatrix}, \text{ où } J_k(\lambda) = \begin{pmatrix} \lambda & 1 & 0 & \ldots & 0 \\ 0 & \lambda & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & \lambda & 1 \\ 0 & \ldots & \ldots & 0 & \lambda \end{pmatrix} \text{ et les } \lambda_i$$

*sont les valeurs propres de A.*

$A$ est **symétrique** si $A^T = A$ ou encore $a_{ij} = a_{ji}$ pour tout $i$ et $j$ de $1, \ldots, n$.

*Si $A \in \mathbb{R}^{n\times n}$ est symétrique, alors ses valeurs propres sont réelles, A est diagonalisable dans $\mathbb{R}$ et on peut choisir une base orthonormée de vecteurs propres ; dans ce cas P est unitaire i.e. $P^{-1} = P^T$ et $P^TAP = D$.*

$A \in \mathbb{R}^{n\times n}$ est **définie positive** si pour tout $X \in \mathbb{R}^n$, $X^T AX \geqslant 0$ et $X^T AX = 0 \Rightarrow X = 0$. Si $A$ est définie positive alors $A$ est inversible.

*Méthode de Gauss : Si $A \in \mathbb{R}^{n\times n}$ (ou $\mathbb{C}^{n\times n}$) est inversible, il existe une matrice de permutation (inversible) $P$ et 2 matrices $L$ et $U$, $L$ étant triangulaire inférieure à diagonale unité, $U$ étant triangulaire supérieure, telles que $PA = LU$.* Le système $AX = b$ se transforme en $LUX = Pb$ et on résout successivement $LY = Pb$ puis $UX = Y$.

*Factorisation de Cholesky : Si $A \in \mathbb{R}^{n\times n}$ est symétrique et définie positive, il existe une unique matrice $C$ triangulaire supérieure à coefficients diagonaux strictement positifs telle que $A = C^T C$.*

S'il est impossible de donner un rappel complet du cours sur les matrices en quelques lignes, on retrouvera aussi des propriétés dans les chapitres suivants : changement de base, localisation des valeurs propres dans les disques de Gershgorin, matrices tridiagonales, matrices à diagonale dominante, méthodes itératives de résolution des systèmes (problème 11.1) etc.

## ÉNONCÉS DES EXERCICES

Le premier paragraphe ci-dessous est une liste de petits exercices qui, sans faire le tour de la question, permet de se (re)familiariser avec le calcul matriciel. Dans le deuxième, on utilise `Matlab` pour diagonaliser une matrice et calculer ses puissances. Enfin le dernier illustre la méthode de la puissance qui permet généralement d'approcher la plus grande valeur propre en module. Il s'agit à nouveau de découvrir un certain nombre d'outils de `Matlab`.

### 2.1 Premiers calculs

**1. Inversion**

Soit $A = \begin{pmatrix} 1 & 2 & 0 & \dots & 0 \\ 0 & 1 & 2 & 0 & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ \vdots & & 0 & 1 & 2 \\ 0 & \dots & \dots & 0 & 1 \end{pmatrix} \in \mathbb{R}^{n\times n}$.

Montrer que $A^{-1} = \begin{pmatrix} 1 & -2 & 4 & \dots & (-2)^{j-1} & \dots & (-2)^{n-1} \\ 0 & 1 & -2 & 4 & \dots & \dots & (-2)^{n-2} \\ & \ddots & \ddots & \ddots & \ddots & & \vdots \\ \vdots & & 0 & 1 & (-2)^{j-i} & \dots & (-2)^{n-i} \\ & & & \ddots & \ddots & \ddots & \vdots \\ & & & & 0 & 1 & -2 \\ 0 & & \dots & & 0 & 0 & 1 \end{pmatrix}$.

**2.** **Matrice triangulaire inversible**

Si $A \in \mathbb{R}^{n\times n}$ est inversible et triangulaire supérieure, montrer que $A^{-1}$ est triangulaire supérieure. Préciser la diagonale.

**3.** **Inverse d'une matrice et valeurs propres**

Montrer que si $A \in \mathbb{R}^{n\times n}$ est inversible, les valeurs propres de $A^{-1}$ sont les inverses des valeurs propres de $A$.

**4.** **Décomposition de Cholesky**

Montrer que la matrice $A = \begin{pmatrix} 4 & 0 & 2 & 0 \\ 0 & 4 & 0 & 2 \\ 2 & 0 & 5 & 0 \\ 0 & 2 & 0 & 5 \end{pmatrix}$ est définie positive. Déterminer la décomposition de Cholesky. Vérifier avec `Matlab`. Tester aussi la décomposition de Schur avec `Matlab`.

**5.** **Matrice définie positive et valeurs propres**

Montrer que si $A \in \mathbb{R}^{n\times n}$ est symétrique et définie positive, ses valeurs propres sont strictement positives.

**6.** **Perturbation du second membre d'un système**

Soient $A = \begin{pmatrix} 1 & 0 \\ 1 & \varepsilon \end{pmatrix}$ et $\Delta A = \begin{pmatrix} 0 & 0 \\ -\varepsilon & \varepsilon^2 \end{pmatrix}$ où $\varepsilon = 10^{-10}$ ; $\Delta A$ est une perturbation de $A$. Si $b = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$, on définit $X$ et $X + \Delta X$ par $AX = b$ et $(A + \Delta A)(X + \Delta X) = b$. Calculer $X$ et montrer que $\Delta X \simeq \begin{pmatrix} 0 \\ 1 \end{pmatrix}$.

Des détails sur ces perturbations sont donnés au chapitre suivant.

## 2.2 Valeurs propres et puissances, un exemple

Soit la matrice $A = \begin{pmatrix} 0 & \frac{1}{4} & \frac{3}{4} \\ \frac{3}{4} & 0 & \frac{1}{4} \\ \frac{1}{4} & \frac{3}{4} & 0 \end{pmatrix}$. À l'aide de `Matlab`,

**1.** Déterminer les valeurs propres et les vecteurs propres de $A$.

Les noms anglais sont eigenvalues et eigenvectors... et on peut consulter l'aide en ligne.

**2.** Construire une matrice de passage qui rend $A$ diagonale.

**3.** Calculer l'inverse de la matrice de passage.

**4.** Vérifier la diagonalisation en effectuant des produits matriciels.

**5.** En déduire $\lim_{n\to\infty} A^n$.

**6.** Le vérifier en calculant $A^{100}$ directement.

## 2.3 Méthode de la puissance

**Rappel :** Nous supposons que $A \in \mathbb{R}^{n\times n}$ est diagonalisable dans $\mathbb{R}$ ou $\mathbb{C}$ et que ses valeurs propres vérifient $|\lambda_1| > |\lambda_2| \geqslant \ldots \geqslant |\lambda_n|$. À noter que $\lambda_1 \in \mathbb{R}$ ; sinon $\bar{\lambda}_1$ serait aussi valeur propre avec $|\lambda_1| = |\bar{\lambda}_1|$. Nous choisissons la norme 2 sur $\mathbb{R}^n$ ou $\mathbb{C}^n$ notée $\|.\|$ i.e $\|X\| = \sum_{i=1}^{n} |x_i|^2$.

Soit $u_1, \ldots, u_n$ une base de vecteurs propres. Nous partons d'un vecteur $x_0$ qui ne soit pas dans le sous-espace engendré par $u_2, \ldots, u_n$. À l'aide de l'algorithme ci-dessous, nous construisons les suites $(\mu_i)$, $(x_i)$, $(q_i)$ :

$$\begin{aligned} q_0 &= \frac{x_0}{\|x_0\|} \\ \text{Pour} \quad i &= 1, 2, \ldots, \\ x_i &= Aq_{i-1}, \\ q_i &= \frac{x_i}{\|x_i\|}, \\ \mu_i &= q_i^T A q_i \end{aligned}$$

Alors la suite $(\mu_i)$ converge vers $\lambda_1$.

**1.** Partant d'une matrice aléatoire $A$ de dimension 10 et d'un vecteur $x_0$ aléatoire, programmer l'algorithme. Prévoir $maxiter$ étapes. Enregistrer le programme sous `puiss1.m`. Test $maxiter = 20$.

**2.** Ajouter à ce programme le calcul de la valeur propre de plus grand module par `Matlab` à l'aide des fonctions `eig, abs, max`. Enregistrer le programme sous `puiss2.m`. Même données test.

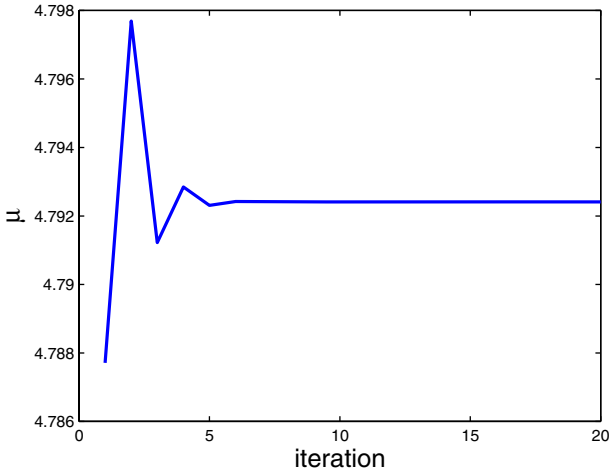
**3.** Ajouter un graphe représentant $\mu(i)$ en fonction de $i$. Enregistrer le programme sous `puiss3.m`. Même données test.

```
>> puiss3
lambda1 =
    4.7924
```

(Cf. figure page suivante.)

Du fait du tirage aléatoire, la valeur propre de module maximum et la figure seront différentes.

**4.** Remplacer la matrice aléatoire par $A_1 = \begin{pmatrix} 1 & 1 & -2 \\ 0 & 1 & -1 \\ 0 & 0 & -0.99 \end{pmatrix}$ puis $A_2 = \begin{pmatrix} 1 & 1 & -2 \\ 0 & -1 & -1 \\ 0 & 0 & 0.1 \end{pmatrix}$ avec maxiter=100.




Pour $A_1$, la suite $(\mu_i)$ semble encore converger vers $\lambda_1$, ce n'est pas le cas pour $A_2$. Les hypothèses ne sont pas vérifiées dans les deux cas ; le premier semble montrer que les conditions de convergence de la suite $(\mu_i)$ sont seulement suffisantes.

## DU MAL À DÉMARRER ?

### 2.1 Premiers calculs

**1.** On peut commencer par une matrice $4 \times 4$ et calculer le produit des deux matrices.

**2.** On peut commencer par une matrice $2 \times 2$ puis procéder par récurrence.

**3.** Écrire $AX = \lambda X$.

**4.** Calculer $\begin{pmatrix} x_1 & x_2 & x_3 & x_4 \end{pmatrix} A \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}$. Pour la décomposition, écrire $A = C^T C$ et procéder par ligne de $C$ ou par colonne.

### 2.2 Valeurs propres et puissances

**2.** À l'aide de l'éditeur, on peut créer le fichier `mat1.m` qui regroupe les commandes. Il suffit ensuite de le lancer en tapant `>> mat1` dans l'interpréteur.

## CORRIGÉS DES EXERCICES

### 2.1 Premiers calculs

**1. Inversion**

$$\begin{pmatrix} 1 & 2 & 0 & \dots & 0 \\ 0 & 1 & 2 & 0 & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ \vdots & & 0 & 1 & 2 \\ 0 & \dots & \dots & 0 & 1 \end{pmatrix} \times \begin{pmatrix} 1 & -2 & 4 & \dots & (-2)^{j-1} & \dots & (-2)^{n-1} \\ 0 & 1 & -2 & 4 & \dots & \dots & (-2)^{n-2} \\ & \ddots & \ddots & \ddots & \ddots & & \vdots \\ \vdots & & 0 & 1 & (-2)^{j-i} & \dots & (-2)^{n-i} \\ & & & \ddots & \ddots & \ddots & \vdots \\ & & & & 0 & 1 & -2 \\ 0 & & \dots & & 0 & 0 & 1 \end{pmatrix} = I.$$

**2. Matrice triangulaire inversible**

Si $A = (a_{i,j})$ est inversible et triangulaire supérieure, ses coefficients diagonaux sont non nuls. Notons $A^{-1} = B = (b_{ij})$. Puisque $BA = I$, nous avons $\sum_{k=1}^{n} b_{i,k}a_{k,j} = \delta_{ij}$. Remarquons que puisque la matrice est triangulaire supérieure, $a_{k,j} = 0$ si $k > j$ si bien que

$$\sum_{k=1}^{j} b_{i,k}a_{k,j} = \delta_{ij}. \tag{2.1}$$

Nous allons montrer par récurrence sur $j$ que $b_{ij} = 0$ si $i > j$ et $b_{jj} = 1/a_{jj}$.

Pour $j = 1$, (2.1) devient $b_{i,1}a_{1,1} = \delta_{i1}$ qui permet d'obtenir $b_{i,1} = 0$ si $i \neq 1$ et $b_{1,1} = 1/a_{1,1}$.

Supposons que pour $j$ donné inférieur ou égal à $n - 1$ et pour tout $k \leqslant j$, $b_{i,k} = 0$ dès que $i > k$. Pour $i > j + 1$, $b_{i,k} = 0$ pour $k = 1, \dots, j$, si bien que (2.1) $b_{i,j+1}a_{j+1,j+1} = \delta_{ij+1}$. Comme $a_{j+1,j+1} \neq 0$, on en déduit que $b_{i,j+1} = 0$ si $i > j + 1$ et $b_{j+1,j+1} = 1/a_{j+1,j+1}$.

Ce qui achève la démonstration.

**3. Inverse d'une matrice et valeurs propres**

Si $\lambda$ est une valeur propre de $A$ de vecteur propre associé $V$, alors $AV = \lambda V$. Si $A$ est inversible, on multiplie cette équation à gauche par $A^{-1}$, il vient $V = A^{-1}\lambda V = \lambda A^{-1}V$. Puisque $\lambda$ est non nul, on divise par $\lambda$ et on obtient que $V$ est vecteur propre de $A^{-1}$ pour la valeur propre $1/\lambda$.

### 4. Décomposition de Cholesky

Soit $X = (x_1, x_2, x_3, x_4)^T$, alors

$$\begin{aligned} X^T AX &= \begin{pmatrix} x_1 & x_2 & x_3 & x_4 \end{pmatrix} \begin{pmatrix} 4 & 0 & 2 & 0 \\ 0 & 4 & 0 & 2 \\ 2 & 0 & 5 & 0 \\ 0 & 2 & 0 & 5 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} \\ &= \begin{pmatrix} x_1 & x_2 & x_3 & x_4 \end{pmatrix} \begin{pmatrix} 4x_1 + 2x_3 \\ 4x_2 + 2x_4 \\ 2x_1 + 5x_3 \\ 2x_2 + 5x_4 \end{pmatrix} \\ &= 4x_1^2 + 4x_1x_3 + 5x_3^2 + 4x_2^2 + 4x_2x_4 + 5x_4^2 \\ &= (2x_1 + x_3)^2 + 4x_3^2 + (2x_2 + x_4)^2 + 4x_4^2 \end{aligned}$$

Donc $X^T AX \geqslant 0$ et si $X^T AX = 0$, alors $2x_1 + x_3 = 0$, $x_3 = 0$, $2x_2 + x_4 = 0$, $x_4 = 0$, système dont l'unique solution est $X = 0$. La matrice est définie positive.

Déterminons $C$ telle que $C = \begin{pmatrix} c_{11} & c_{12} & c_{13} & c_{14} \\ 0 & c_{22} & c_{23} & c_{24} \\ 0 & 0 & c_{33} & c_{34} \\ 0 & 0 & 0 & c_{44} \end{pmatrix}$ et $C^T C = A$.

Remarquons que la première colonne de $C^T C$ est $\begin{pmatrix} c_{11}^2 \\ c_{12}c_{11} \\ c_{13}c_{11} \\ c_{14}c_{11} \end{pmatrix}$. Sachant que $c_{11} > 0$, on en déduit $c_{11} = 2$, $c_{12} = 0$, $c_{13} = 1$, $c_{14} = 0$. La première ligne de $C$ est déterminée.

À l'aide de ces informations, la deuxième colonne de $C^T C$ est

$$\begin{pmatrix} 2 \times 0 \\ 0^2 + c_{22}^2 \\ 1 \times 0 + c_{23}c_{22} \\ 0 \times 0 + c_{24}c_{22} \end{pmatrix} = \begin{pmatrix} 0 \\ 4 \\ 0 \\ 2 \end{pmatrix}.$$

Puisque $c_{22} > 0$, on en déduit $c_{22} = 2$, $c_{23} = 0$, $c_{24} = 1$. On remarque aussi qu'il suffisait de s'intéresser aux trois dernières lignes $C^T C$.

Dans la troisième colonne, les deux dernières lignes vérifient $\begin{pmatrix} 1 + c_{33}^2 \\ c_{34}c_{33} \end{pmatrix} = \begin{pmatrix} 5 \\ 0 \end{pmatrix}$. Puisque $c_{33} > 0$, on en déduit $c_{33} = 2$, $c_{34} = 0$.

Enfin la dernière ligne multipliée par la dernière colonne donne $1 + c_{44}^2 = 5$, soit $c_{44} = 2$.

```
>> A=[4 0 2 0;0 4 0 2;2 0 5 0; 0 2 0 5];
>> T=chol(A)
T =
     2     0     1     0
     0     2     0     1
     0     0     2     0
     0     0     0     2
>> [U,S] = schur(A)
U =
   -0.7882         0         0   -0.6154
         0    0.7882    0.6154         0
    0.6154         0         0   -0.7882
         0   -0.6154    0.7882         0

S =
    2.4384         0         0         0
         0    2.4384         0         0
         0         0    6.5616         0
         0         0         0    6.5616
```

**5. Matrice définie positive et valeurs propres**

Si $A \in \mathbb{R}^{n\times n}$ est symétrique, ses valeurs propres sont réelles. Soit $\lambda$ une valeur propre et $X \in \mathbb{R}^n$ un vecteur propre associé. On a donc $AX = \lambda X$ soit en multipliant par $X^T$, $X^T AX = \lambda X^T X$. Si $A$ est définie positive, alors puisque $X \neq 0$, $X^T AX > 0$. Par ailleurs $X^T X = \sum_{i=1}^{n} x_i^2 > 0$, donc $\lambda > 0$.

**6. Perturbation du second membre d'un système**

Si $\begin{pmatrix} 1 & 0 \\ 1 & \varepsilon \end{pmatrix} X = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$, alors $X = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ et si $\begin{pmatrix} 1 & 0 \\ 1-\varepsilon & \varepsilon+\varepsilon^2 \end{pmatrix} (X+\Delta X) = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$, alors $X+\Delta X = \begin{pmatrix} 1 \\ 1/(1+\varepsilon) \end{pmatrix}$, soit $\Delta X \simeq \begin{pmatrix} 0 \\ 1 \end{pmatrix}$.

Une petite perturbation de la matrice peut entraîner une perturbation importante de la solution du système.

## 2.2 Valeurs propres et puissances, un exemple

Le programme ci-dessous calcule les valeurs propres de $A$ qui sont donc $\lambda_1 = 1$, $\lambda_2 = -0.5000 + 0.4330i$, $\lambda_3 = \bar{\lambda}_2$ aux erreurs d'arrondi près. Donc $A$ est diagonalisable et il existe $P \in \mathbb{C}^{3\times 3}$ telle que $A = P^{-1}DP$. On en déduit que $A^n = PD^nP^{-1}$. Puisque pour $i = 2, 3$, on a $|\lambda_i| < 1$,

$$\lim_{n\leftarrow+\infty} D^n = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \text{ et } \lim_{n\leftarrow+\infty} A^n = P \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} P^{-1}.$$

mat1.m

```
A=[0 1/4 3/4;3/4 0 1/4;1/4 3/4 0]
[P,D] = eig(A)
W=inv(P);
W*A*P
limAn=P*[1 0 0;0 0 0;0 0 0]*W
A^100
```

## 2.3 Méthode de la puissance

puiss3.m

```
maxiter=20;tol=1e-5;
A=rand(10);
%A=[1 1 -2;0 1 -1;0 0 -0.99];
%A=[1 1 -2;0 -1 -1;0 0 0.1];

X=rand(length(A),1);
Q=X/norm(X);
for i=1:maxiter
    X=A*Q;
    Q=X/norm(X);
    mu(i)=Q'*A*Q;
end;
valp=eig(A);
[lambda1,i]=max(abs(valp));
lambda1= valp(i)
plot(1:maxiter,mu)
xlabel('iteration','FontSize',16)
ylabel('\mu','FontSize',16)
```

### Ce qu'il faut retenir de cet exercice

On notera la boucle sans test de convergence, ce qui n'est pas satisfaisant. À défaut d'un vrai test de convergence, on peut contrôler la différence entre deux itérés avec la boucle :

```
tol = 1 e-10;
j=1;
while (erreur>tol)&(j<=maxiter)
    X=A*Q;
    Q1=X/norm(X);
    mu(j)=Q1'*A*Q1;
    erreur=norm(Q-Q1);
    Q=Q1;
    j=j+1;
end;
if (j>maxiter) disp('methode␣non␣convergente'); end;
```

# 3

# Matrices, normes et conditionnement

## RAPPEL DE COURS

Nous vous proposons d'étudier les normes matricielles et le **conditionnement** qui permet de majorer les variations de la solution d'un système linéaire quand on modifie le second membre ou la matrice. Rappelons que si $\|\cdot\|$ est une norme sur $\mathbb{R}^n$ (ou $\mathbb{C}^n$) alors la **norme matricielle subordonnée** est définie pour $A \in \mathbb{R}^{n\times n}$ (ou $\mathbb{C}^{n\times n}$) par $\|A\| = \sup_{X\neq 0} \dfrac{\|AX\|}{\|X\|}$. Par linéarité, elle coïncide avec $\sup_{\|X\|=1} \|AX\|$. *Alors* $\|AB\| \leqslant \|A\| \times \|B\|$ *pour A matrice et B vecteur ou matrice.*

Pour les normes usuelles de $\mathbb{R}^n$ (ou $\mathbb{C}^n$), si $X = (x_1, \ldots, x_n)^T$ et $A = (a_{ij}) \in \mathbb{R}^{n\times n}$ (ou $\mathbb{C}^{n\times n}$), on montre (excellents exercices) que

- *Si* $\|X\|_1 = \sum_{i=1}^{n} |x_i|$ *alors* $\|A\|_1 = \max_{j=1,\ldots,n} \sum_{i=1}^{n} |a_{ij}|$,
- *Si* $\|X\|_\infty = \max_{i=1,\ldots,n} |x_i|$ *alors* $\|A\|_\infty = \max_{i=1,\ldots,n} \sum_{j=1}^{n} |a_{ij}|$,
- *Si* $\|X\|_2 = \left(\sum_{i=1}^{n} |x_i|^2\right)^{1/2}$ *alors* $\|A\|_2 = \left(\rho(A^*A)\right)^{1/2}$ *avec* $A^* = \bar{A}^T$, *où* $\rho(B)$ *désigne le rayon spectral de B, c'est-à-dire la plus grande valeur propre en module.*

Enfin, pour une norme donnée, le conditionnement d'une matrice $A \in \mathbb{R}^{n\times n}$ (ou $\mathbb{C}^{n\times n}$) inversible est défini par $\text{cond}(A) = \|A\| \times \|A^{-1}\|$.

*Si* $A \in \mathbb{R}^{n\times n}$ *(ou* $\mathbb{C}^{n\times n}$*) est inversible, si b et* $\delta b$ *sont deux vecteurs de* $\mathbb{R}^n$ *(ou* $\mathbb{C}^n$*) et que* $AX = b$, $A(X + \delta X) = b + \delta b$, *alors*

$$\frac{\|\delta X\|}{\|X\|} \leqslant \text{cond}(A)\frac{\|\delta b\|}{\|b\|}. \tag{3.1}$$

Démonstration dans l'exercice 3.2.

*Si $A$, $A+\Delta A \in \mathbb{R}^{n\times n}$ (ou $\mathbb{C}^{n\times n}$) sont inversibles, si $b$ est un vecteur de $\mathbb{R}^n$ (ou $\mathbb{C}^n$) et que $AX = b$, $(A+\delta A)(X+\Delta X) = b$, alors*

$$\frac{\|\Delta X\|}{\|X+\Delta X\|} \leqslant \mathrm{cond}(A)\frac{\|\Delta A\|}{\|A\|}. \tag{3.2}$$

## ÉNONCES DES EXERCICES

Dans ce chapitre, après quelques calculs élémentaires, nous avons volontairement mêlé les exercices théoriques et les utilisations de `Matlab`. Il s'agit parfois de vérifier numériquement un résultat déjà démontré ou encore de calculer numériquement avant de démontrer, ce qui est souvent la démarche en analyse numérique. Le dernier exercice est plus classique dans sa démarche ; une méthode numérique est rapidement décrite puis programmée.

Le premier exercice calcule une norme de matrice, puis compare norme et rayon spectral. Le second relie conditionnement et erreur et les deux suivants montrent qu'il n'y a pas de lien direct entre conditionnement et valeurs propres ou déterminant. Le quatrième permet de revenir sur une inégalité obtenue dans le premier exercice qui peut systématiquement se transformer en égalité. Le dernier exercice prend un peu d'avance sur les techniques de moindres carrés mais permet, sans doute, d'illustrer de manière pertinente la question du conditionnement.

### 3.1 Premiers calculs

**1.** **Calcul de norme**

Soit $A = \begin{pmatrix} 1 & 2 & 0 & \dots & 0 \\ 0 & 1 & 2 & 0 & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ \vdots & & 0 & 1 & 2 \\ 0 & \dots & \dots & 0 & 1 \end{pmatrix} \in \mathbb{R}^{n\times n}$. Déterminer $\|A\|_1$, $\|A^{-1}\|_1$ et $\mathrm{cond}_1(A)$.

**2.** **Norme et rayon spectral**

Soit $\|.\|$ une norme sur $\mathbb{R}^n$. Si $A \in \mathbb{R}^{n\times n}$, montrer que $\rho(A) \leqslant \|A\|$
où $\rho(A) = \max\{|\lambda_i| : \lambda_i \text{ valeur propre de } A\}$.

### 3.2 Conditionnement et erreur

On choisit une norme sur $\mathbb{R}^n$. Soient $b$ et $\delta b$ 2 vecteurs de $\mathbb{R}^n$ avec $b \neq 0$ et $A = (a_{ij}) \in \mathbb{R}^{n\times n}$ une matrice inversible. Puisque $A$ est inversible, soient $X$ l'unique solution de $AX = b$ et $X + \delta X$ la solution de $A(X+\delta X) = b + \delta b$.

**1.** Montrer que $\|b\| \leqslant \|A\| \times \|X\|$ puis que $\|\delta X\| \leqslant \|A^{-1}\| \times \|\delta b\|$.

**2.** En déduire que :

$$\frac{\|\delta X\|}{\|X\|} \leqslant \text{cond}(A)\frac{\|\delta b\|}{\|b\|}.$$

On majore ainsi l'erreur relative sur la solution en fonction de l'erreur relative sur le second membre.

**3.** Le pire (cas d'égalité) peut parfois arriver. Ainsi dans l'exemple proposé par R.S. Wilson et repris dans [18]

$$A = \begin{pmatrix} 10 & 7 & 8 & 7 \\ 7 & 5 & 6 & 5 \\ 8 & 6 & 10 & 9 \\ 7 & 5 & 9 & 10 \end{pmatrix},\ b = \begin{pmatrix} 32 \\ 23 \\ 33 \\ 31 \end{pmatrix},\ \delta b = \begin{pmatrix} 0,01 \\ -0,01 \\ 0,01 \\ -0,01 \end{pmatrix}.$$

(a) À l'aide de `Matlab`, résoudre les systèmes $AX = b$ et $AY = b + \delta b$.

(b) Évaluer numériquement $\|Y - X\|_\infty / \|X\|_\infty$, $\text{cond}_\infty(A)$, $\|Y - X\|_\infty / \|X\|_\infty - \text{cond}_\infty(A).\|\delta b\|_\infty / \|b\|_\infty$.

(c) Mêmes questions avec $\|\cdot\|_2$.

## 3.3 Conditionnement et valeurs propres

On définit $f(x_0, ..., x_n) = \displaystyle\int_0^1 (x_0 + x_1 t + ... + x_n t^n)^2 dt$, $x_i \in \mathbb{R}$, $i = 0, \ldots, n$.

**1.** Montrer que $f(x_0, ..., x_n) = X^T A X$ avec

$$X^T = (x_0, ..., x_n) \text{ et } A = (a_{ij})_{0 \leqslant i,j \leqslant n} \text{ où } a_{ij} = \frac{1}{1 + i + j}$$

**2.** En déduire que $A$ est une matrice symétrique, définie positive.

**3.** Montrer que si $\lambda$ est valeur propre de $A$ alors $\lambda$ est réelle et pour toute norme sur $\mathbb{R}^{n+1}$,

$$\frac{1}{\|A^{-1}\|} \leqslant \lambda \leqslant \|A\|. \tag{3.3}$$

**4.** Pour un $n$ fixé, à l'aide de `Matlab` (et sans boucle) construire $A$ et calculer $A^{-1}$ .

**5.** Calculer numériquement $\|A\|_\infty$ et $\|A^{-1}\|_\infty$.

```
n =
     5
nA =
    2.4500
ninvA =
  1.1865e+007
```

**6.** Majorer les valeurs propres à l'aide de (3.3) et calculer numériquement le conditionnement de $A$ pour $\|\cdot\|_\infty$.

```
conditionnement =
  2.9070e+007
```

**7.** Avec $n = 5$, pour résoudre $AX = b$, on connaît $b$ avec 5 chiffres significatifs (c'est-à-dire $\frac{\|\delta b\|_\infty}{\|b\|_\infty} \leqslant 10^{-5}$ ). Avec combien de chiffres significatifs est-on assuré de connaître $X$ (c'est-à-dire $\frac{\|\delta X\|_\infty}{\|X\|_\infty}$) ?

## 3.4 Conditionnement et déterminant

Soient $E$ et $U$ des matrices carrées d'ordre $n$ et $f \in \mathbb{R}^n$ définis par

$$E = \begin{pmatrix} 0 & \cdots & \cdots & \cdots & 0 \\ 1 & 0 & \cdots & & 0 \\ 0 & 1 & 0 & & \vdots \\ \vdots & \ddots & \ddots & \ddots & \\ 0 & \cdots & 0 & 1 & 0 \end{pmatrix}, \; U = \begin{pmatrix} 1 & \cdots & \cdots & \cdots & 1 \\ 0 & 1 & \cdots & \cdots & 1 \\ \vdots & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \vdots \\ 0 & \cdots & \cdots & 0 & 1 \end{pmatrix}, \; f = \begin{pmatrix} 1 \\ 1 \\ \vdots \\ 1 \\ 1 \end{pmatrix}$$

et

$$A = \begin{pmatrix} 2 & 2 & \cdots & \cdots & 2 \\ -1 & 1 & 1 & \cdots & 1 \\ 0 & -1 & 1 & \cdots & 1 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & -1 & 1 \end{pmatrix}, \; I \text{ est la matrice identité}$$

**1.** Vérifier que $A = U - E + e_1 f^T$ où $e_1$ est le premier vecteur de la base canonique.

**2.** Montrer que $A = (2I - E) \times U$. En déduire $\det(A)$.

**3.** Calculer $E^2$, $E^3$,..., $E^n$.

**4.** Montrer que $U^{-1} = I - E^T$ puis que $(I - \frac{E}{2})^{-1} = I + \frac{E}{2} + \frac{E^2}{4} + ... + \frac{E^{n-1}}{2^{n-1}}$.

**5.** En déduire $A^{-1}$ et montrer que $\|A^{-1}\|_\infty = 1 - \dfrac{1}{2^n}$.

**6.** Montrer alors que $\text{cond}_\infty(A) \sim 2n$.

## 3.5 Norme 2

**1.** À l'aide de `Matlab` construire une matrice carrée aléatoire $A$ de dimension 10.

**2.** Calculer $S = A^T \times A$ et déterminer les valeurs propres de $S$. (Montrer directement que les valeurs propres de $S$ sont réelles et positives ou nulles). Déterminer la plus grande valeur propre $\lambda$ de $S$ et la plus petite $\mu$. Comparer $\|A\|_2$ et $\sqrt{\lambda}$ puis $\|A^{-1}\|_2$ et $\dfrac{1}{\sqrt{\mu}}$. Faire plusieurs tests.

On a ainsi vérifié numériquement le calcul de la norme matricielle $\|\cdot\|_2$.

**3.** À partir de la matrice $A$ précédente ou d'une autre matrice aléatoire :

(a) On construit $b = AX$ où $X = (1 \ldots 1)^T$.

(b) On construit un vecteur aléatoire $\delta b$ et on détermine $\delta X$ tel que $A(X + \delta X) = b + \delta b$.

(c) Comparer numériquement $\dfrac{\|\delta b\|_2}{\|b\|_2}$, $\dfrac{\|\delta X\|_2}{\|X\|_2}$ et $\|A\|_2 \times \|A^{-1}\|_2 \times \dfrac{\|\delta b\|_2}{\|b\|_2}$.

On a ainsi vérifié numériquement l'inégalité (3.1).

**4.** Le pire arrive..., cas d'égalité.

On reprend la matrice précédente et $S = A^T \times A$. La plus grande valeur propre de $S$ étant $\lambda$, soit $U$ un vecteur propre associé. De même le couple $(\mu, V)$ correspond à la plus petite valeur propre et à un vecteur propre associé. On définit $b = AU$ et $\delta b = AV$, donc $X = U$ et $\delta X = V$.

Comparer $\dfrac{\|\delta X\|_2}{\|X\|_2}$ et $\|A\|_2 \times \|A^{-1}\|_2 \times \dfrac{\|\delta b\|_2}{\|b\|_2}$ ; on pourra utiliser les instructions `[V,D] = eig(S)` et `[C,I] = max(...)` qui précise la position `I` du maximum. Tester sur plusieurs exemples.

**5.** Démontrer dans ce cas l'égalité

$$\frac{\|\delta X\|_2}{\|X\|_2} = \text{cond}(A)\frac{\|\delta b\|_2}{\|b\|_2}.$$

## 3.6 Approximation polynômial

Cet exercice pourrait être placé dans le chapitre 6 ; on pourra d'ailleurs commencer par en étudier le premier exercice. Il s'agit d'une méthode de moindres carrés.

➤ Introduction

Soit $f$ une fonction continue sur $[0,1]$. On cherche à approcher $f$ par une suite de polynômes de degré $n$. Pour $n$ donné, on minimise

$$E(p) = \int_0^1 \big(f(t) - p(t)\big)^2 \sqrt{t}\, dt,\ p \in \mathbb{R}_n[X].$$

Si $p_n(x) = \sum_{i=0}^{n} a_i x^i$ , le minimum est atteint lorsque $\dfrac{\partial E}{\partial a_i} = 0$ pour $i = 0$ à $n$ ce qui donne le système :

$$M\alpha = b$$

où

$$M = (m_{ij})_{i=0,\ldots,n,\ j=0,\ldots,n},\ \text{avec } m_{ij} = \int_0^1 t^{i+j+1/2} dt = \frac{1}{i+j+3/2},$$
$$b = (b_i)_{i=0,\ldots,n},\ \text{avec } b_i = \int_0^1 f(t) t^{i+1/2} dt,$$

$$\alpha = (a_0, a_1, \ldots, a_n)^T, \text{ coefficients de } p_n.$$

➤ Programmation

**1.** $f$ est donnée par un fichier `f1.m` que vous construisez. Exemple $f1(x) = \sin \pi x$. Calculer `f1([1/4 -1])`.

```
>> f1([1/4 -1])
ans =
    0.7071   -0.0000
```

**2.** Construire la matrice $M \in \mathbb{R}^{(n+1)\times(n+1)}$ ; attention les indices `Matlab` vont de 1 à $n+1$. Sauvegarde sous `approx1.m`. Test avec $n = 4$.

```
>> approx1
M =
    0.6667    0.4000    0.2857    0.2222    0.1818
    0.4000    0.2857    0.2222    0.1818    0.1538
    0.2857    0.2222    0.1818    0.1538    0.1333
    0.2222    0.1818    0.1538    0.1333    0.1176
    0.1818    0.1538    0.1333    0.1176    0.1053
```

**3.** Construire le second membre à l'aide des fichiers `sndmemb.m` et `fonctj.m` ci-dessous ; l'appel se fait par `b=sndmemb('f',n)` où `f` est le nom du fichier contenant la fonction. Ces fichiers permettent de calculer une approximation de l'intégrale à l'aide de la fonction `quadl.m`. Test avec `f=f1` et $n = 4$.

```
function y=fonctj(x,nomf,j);
y=feval(nomf,x).*x.^(j+1/2);
```

```
function b=sndmemb(nomf,n);
for i=1:n+1
      b(i)=quadl('fonctj',0,1,[ ],[ ],nomf,i-1);
end;
```

```
>> sndmemb('f1',4)
ans =
    0.4374    0.2416    0.1521    0.1041    0.0755
```

**4.** Une fois la solution $\alpha$ déterminée, le polynôme se calcule à l'aide de `polyval` (attention à l'ordre des coefficients). Calculer $\alpha$ et tracer le graphe de $f$ puis celui du polynôme $p_n$. Sauvegarde dans `approx2.m`. Test avec $n = 4$, `f1`.

```
>> approx2
Coefficients du polynome
a =
    0.0022
    3.0765
    0.5720
   -7.2921
    3.6428
```

**5.** Faire varier $n$. Changer la fonction. Que se passe-t-il quand $n$ augmente ($n \geqslant 20$) ?

**Exemples** $f1(x) = \sin \pi x$ ; $f2(x) = \sin 4\pi x$ ; $f3(x) = |x-1/2|$ ; $n = 2, 3, 5, 10, 15, 20...$

## DU MAL À DÉMARRER ?

### 3.1 Premiers calculs

**1.** À noter que $A^{-1}$ est donnée dans l'exercice 2.1.

**2.** Si $U$ vecteur propre, écrire $\lambda U = AU$ et majorer la norme.

### 3.2 Conditionnement et erreur

Utiliser la forme $f(x_0, \ldots, x_n) \geqslant 0$.

### 3.4 Conditionnement et déterminant

**1.** Commencer par des matrices $4 \times 4$.

**2.** Pour montrer $N = M^{-1}$, on peut montrer que $MN = I$

### 3.5 Norme 2

**5.** On pourra commencer par montrer que $b^T b = \lambda X^T X$ en partant de $SX = \lambda X$

## CORRIGÉS DES EXERCICES

### 3.1 Premiers calculs

**1.** $A = \begin{pmatrix} 1 & 2 & 0 & \dots & 0 \\ 0 & 1 & 2 & 0 & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ \vdots & & 0 & 1 & 2 \\ 0 & \dots & \dots & 0 & 1 \end{pmatrix}$ et $A^{-1} = \begin{pmatrix} 1 & -2 & 4 & \dots & (-2)^{j-1} & \dots & (-2)^{n-1} \\ 0 & 1 & -2 & 4 & \dots & \dots & (-2)^{n-2} \\ & \ddots & \ddots & \ddots & \ddots & & \vdots \\ \vdots & & 0 & 1 & (-2)^{j-i} & \dots & (-2)^{n-i} \\ & & & \ddots & \ddots & \ddots & \vdots \\ & & & & 0 & 1 & -2 \\ 0 & & \dots & & 0 & 0 & 1 \end{pmatrix}$.

On a alors $\|A\|_1 = 3$ et $\|A^{-1}\|_1 = 1+2+4+\ldots+2^{n-1} = 2^n - 1$ si bien que $\text{cond}_1(A) = 3.(2^n - 1)$.

**2.** Si $\lambda$ est une valeur propre de $A$ de vecteur propre associé $U$, alors $AU = \lambda U$.
Donc $\|\lambda U\| = |\lambda|.\|U\| \leqslant \|A\|.\|U\|$. Puisque $U \neq 0$, on a $\|U\| > 0$, d'où $|\lambda| \leqslant \|A\|$. La propriété est vérifiée pour toute valeur propre donc en particulier pour la plus grande en module si bien que $\rho(A) \leqslant \|A\|$.

### 3.2 Conditionnement et erreur

Sachant que $AX = b$ et $A(X + \delta X) = b + \delta b$, par différence, on obtient $A\delta X = \delta b$ d'où $\delta X = A^{-1}\delta b$.

$$\begin{aligned} AX = b &\Rightarrow \|b\| \leqslant \|A\| \times \|X\|, \\ \delta X = A^{-1}\delta b &\Rightarrow \|\delta X\| \leqslant \|A^{-1}\| \times \|\delta b\|. \end{aligned}$$

En multipliant les deux inégalités, il vient $\|b\| \times \|\delta X\| \leqslant \|A\| \times \|A^{-1}\| \times \|X\| \times \|\delta b\|$. Sachant que $b \neq 0$ par hypothèse et que $X \neq 0$ sinon on aurait $b = 0$, on déduit la majoration (3.1).

Programme `erreur.m`. On notera la résolution de système qui se fait en une seule commande, ainsi que l'appel aux fonctions `norm` et `cond`.

```
A=[10 7 8 7;7 5 6 5; 8 6 10 9;7 5 9 10];
b=[32 23 33 31]';
deltab=[0.01 -0.01 0.01 -0.01]';
```

```
X=A \b;
Y=A (\b+deltab);
r1=norm(Y-X,inf);
r2=r1/norm(X,inf);
c=cond(A,inf);
r3=r2-c*norm(deltab,inf)/norm(b,inf);
```

On trouve $r3 = -2.1894 \times 10^{-013}$, très proche de 0. L'inégalité (3.1) est donc quasi une égalité. En modifiant le programme (remplacer `norm(.,inf)` par `norm(.,2)`), on s'aperçoit que pour la norme 2, l'inégalité reste stricte ($r3 = -0.1744$).

### 3.3 Conditionnement et valeurs propres

Notons que

$$f(x_0,\ldots,x_n) = \int_0^1 (\sum_{i=0}^{n} x_i t^i)(\sum_{j=0}^{n} x_j t^j)dt = \sum_{i=0}^{n}\sum_{j=0}^{n} x_i x_j \int_0^1 t^{i+j}dt = \sum_{i=0}^{n}\sum_{j=0}^{n} x_i x_j \frac{1}{i+j+1}.$$

Cette dernière expression s'écrit $X^T AX$ où $X = (x_0,\ldots,x_n)^T$ et $A = (a_{ij})_{i,j=0,\ldots,n}$ avec $a_{ij} = \dfrac{1}{i+j+1}$.

Sachant que $a_{ij} = a_{ji}$, A est symétrique. De plus par définition de $f$,

$f(x_0,\ldots,x_n) = X^T AX \geqslant 0$ et si $X^T AX = 0$ alors $\displaystyle\sum_{i=0}^{n} x_i t^i = 0$ (continuité et positivité de la fonction à intégrale nulle). Si le polynôme s'annule sur $[0,1]$, on en déduit que ces coefficients sont nuls donc $X = 0$. Finalement $A$ est définie positive.

Comme $A$ est symétrique, ses valeurs propres sont réelles. De plus elles sont strictement positives puisque $A$ est définie positive. Si $\lambda$ est une valeur propre de vecteur propre associé $X \neq 0$ i.e. $AX = \lambda X$ alors $\dfrac{\|AX\|}{\|X\|} = \lambda$. On en déduit $\lambda \leqslant \|A\|$. Par ailleurs $\dfrac{1}{\lambda}X = A^{-1}X$ et on obtient ainsi $\dfrac{1}{\lambda} \leqslant \|A^{-1}\|$ d'où la double inégalité demandée : $\dfrac{1}{\|A^{-1}\|} \leqslant \lambda \leqslant \|A\|$.

`condvalp.m`. On notera la construction de $A$ en une seule commande.

```
n=5
A=1./((0:n)'*ones(1,n+1)+ones(n+1,1)*(0:n)+1);
invA=A^(-1);
nA=norm(A,inf)
ninvA=norm(invA,inf)
conditionnement=cond(A,inf)
```

Sachant que pour $n = 5$, le conditionnement est $2.9070 \times 10^{+007}$, la précision relative garantie pour $X$ est donc de $2.9 \times 10^2$ puisque celle de $b$ est de l'ordre de $10^{-5}$. En conclusion, même avec des valeurs propres « raisonnables », on peut avoir un très mauvais conditionnement.

## 3.4 Conditionnement et déterminants

Les premières questions sont des calculs sur les matrices. En cas de difficulté, on pourra commencer par des matrices $4 \times 4$ par exemple. Noter que $\det A = \det(2I - E) \times \det U$ et que, comme les matrices sont triangulaires, ce déterminant vaut $2^n$.

Pour montrer les résultats suivants, on calcule $U \times (I - E^T)$ puis
$(I - \frac{E}{2}) \times (I + \frac{E}{2} + \frac{E^2}{4} + ... + \frac{E^{n-1}}{2^{n-1}})$. Dans ce calcul on évaluera les puissances successives de $E$ et on notera en particulier que $E^n = 0$ puisque la sous-diagonale de 1 descend d'un cran à chaque puissance $E^k$.

Puisque $A = 2(I - E/2) \times U$, on en déduit

$$A^{-1} = \frac{1}{2}U^{-1} \times (I - E/2)^{-1} = \frac{1}{2}(I - E^T) \times (I + \frac{E}{2} + \frac{E^2}{4} + ... + \frac{E^{n-1}}{2^{n-1}}).$$

D'où

$$A^{-1} = \frac{1}{2}\begin{pmatrix} 1/2 & -1 & 0 & 0 & \dots & 0 \\ 1/4 & 1/2 & -1 & 0 & \dots & 0 \\ 1/8 & 1/4 & 1/2 & -1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ 1/2^{n-1} & 1/2^{n-2} & \dots & \dots & 1/2 & -1 \\ 1/2^{n-1} & 1/2^{n-2} & \dots & \dots & 1/2 & 1 \end{pmatrix}$$

Ceci nous permet de calculer $\|A\|_\infty = 1 - 1/2^n$ puis $\text{cond}_\infty(A) = 2n.(1 - 2^{-n}) \sim 2n$. On voit donc que le déterminant et le conditionnement ne sont pas du même ordre de grandeur.

## 3.5 Norme 2

**1. 2.** Si $S = A^T A$ alors $S^T = A^T(A^T)^T = A^T A = S$ si bien que $S$ est symétrique. Donc ses valeurs propres sont réelles. Soit $\lambda$ une valeur propre et $X$ un vecteur propre associé. En multipliant $SX = \lambda X$ par $X^T$ à gauche, il vient $X^T A^T AX = \lambda X^T X$ soit $(AX)^T(AX) = \lambda X^T X$. Puis que $X \neq 0$, $X^T X = \sum_{i=0}^{n} x_i^2 > 0$. Par ailleurs $(AX)^T(AX) = \sum_{i=0}^{n}(AX)_i^2 \geqslant 0$ et donc $\lambda \geqslant 0$.

À l'aide du programme ci-dessous, on vérifie numériquement que $\|A\|_2 = \sqrt{\lambda}$ où $\lambda$ est la plus grande valeur propre de $S = A^T \times A$ et de même pour $\|A^{-1}\|_2$.

`valp.m`

```
A=rand(10);
S=A'*A;
Lambda=eig(S)
lmax=max(Lambda)
lmin=min(Lambda)
normA=norm(A,2)
```

```
raclambda=sqrt(lmax)
normAm1=norm(A^(-1),2)
unsurracmu=1/sqrt(lmin)
```

**3.** Le programme suivant permet de vérifier numériquement l'inégalité (3.1).

`erreur.m`

```
A=rand(10);
X=ones(10,1);
b=A*X;
deltab=rand(10,1);
deltaX=A\deltab;
nb=norm(b,2);
ndeltab=norm(deltab,2);
nX=norm(X,2);
ndeltaX=norm(deltaX,2);
rb=ndeltab/nb
rX=ndeltaX/nX
cond(A,2)*rb
```

**4.** Le programme ci dessous permet de déterminer un cas d'égalité dans (3.1). Cette égalité est obtenue aux erreurs machine près, de l'ordre de $10^{-15}$.

`egalite.m`

```
A=rand(10);
S=A'*A;
[P,D]=eig(S);
tlambda=max(D) %tableau des valeurs propres
[lambdamax,imax]=max(tlambda); % valeur propre max et position
[lambdamin,imin]=min(tlambda); % valeur propre min et position
U=P(:,imax);
V=P(:,imin);
b=A*U;
deltab=A*V;
X=U;
deltaX=V;
nb=norm(b,2);
ndeltab=norm(deltab,2);
nX=norm(X,2);
ndeltaX=norm(deltaX,2);
rb=ndeltab/nb
rX=ndeltaX/nX
cond(A,2)*rb-rX
```

### Ce qu'il faut retenir de cet exercice

On notera que pour la fonction `max` de `Matlab`, non seulement on trouve le maximum mais aussi la position de ce maximum.

**5.** Montrons ce cas d'égalité. Rappelons que $\sqrt{\lambda} = \rho(S)^{1/2} = \|A\|_2$ et de même $1/\sqrt{\mu} = \|A^{-1}\|_2$. Nous avons alors $A^T AX = \lambda X$ donc, en multipliant à gauche par $X^T$

$$X^T A^T AX = (AX)^T (AX) = \|AX\|_2^2 = \lambda X^T X = \lambda\big(\|X\|_2\big)^2.$$

Puisque $AX = b$, on en déduit $\sqrt{\lambda} = \dfrac{\|b\|_2}{\|X\|_2}$. De même sachant que $A^T \times A\delta X = \mu\delta X$, on obtient $\sqrt{\mu} = \dfrac{\|\delta b\|_2}{\|\delta X\|_2}$. En divisant la première égalité par la seconde, on déduit le résultat voulu :

$$\frac{\|\delta X\|_2}{\|X\|_2} = \frac{\sqrt{\lambda}}{\sqrt{\mu}} \frac{\|\delta b\|_2}{\|b\|_2} = \text{cond}(A) \frac{\|\delta b\|_2}{\|b\|_2}.$$

## 3.6 Approximation polynômiale

`approx.m`

```
n=4;
nomf='f1';
M=1./((0:n)'*ones(1,n+1)+ones(n+1,1)*(0:n)+3/2);
b=sndmemb(nomf,n);
a=M\b'; %coeff du polynome
disp('Coefficients du polynome')
a
x=[0:0.01:1];
y=feval(nomf,x);
appr=polyval(a(length(a):-1:1),x);
plot(x,y,x,appr');
```

Quand *n* augmente, l'approximation se dégrade, ce qui est théoriquement impossible. En fait, plus *n* augmente, plus la matrice *M* est mal conditionnée ; d'ailleurs `Matlab` le signale. D'autre part, le second membre a été approché. Tout ceci explique les erreurs sur les coefficients $\alpha$ et les graphes catastrophiques des polynômes d'approximation.

### Ce qu'il faut retenir de cet exercice

`Matlab` a précisé dans l'interpréteur que les matrices étaient mal conditionnées tout en poursuivant les calculs. Quand la machine donne des commentaires, c'est à l'utilisateur de savoir en tenir compte...

# 4 Interpolation polynômiale

## RAPPEL DE COURS

Étant donnés $n+1$ couples de réels $(x_i, y_i)$, il s'agit de trouver une fonction $\phi$ telle que $\phi(x_i) = y_i$ pour $i = 0, \ldots, n$. $\phi$ peut être un polynôme (Interpolation de Lagrange), un polynôme trigonométrique ou une fonction régulière polynômiale par morceaux (spline). On désigne par $\mathbb{P}_k$ l'espace des polynômes de degré inférieur ou égal à $k$ ainsi que l'espace des fonctions polynômiales correspondantes.

**Interpolation de Lagrange :** Soient $x_0 < x_2 < \ldots < x_n$, $n+1$ réels distincts et $n+1$ réels $y_i$. *Il existe un unique polynôme $p \in \mathbb{P}_n$ tel que $p(x_i) = y_i$ pour $i = 0, \ldots, n$.*

On suppose que $y_i = f(x_i)$ où $f$ est une fonction définie et de classe $\mathcal{C}^{n+1}$ sur un intervalle fermé $I = [a, b]$ contenant tous les $x_i$. Soit $x \in I$. Alors *il existe $\xi$ appartenant au plus petit intervalle ouvert contenant $x$ et les $x_i$ tel que*

$$f(x) - p(x) = \frac{1}{(n+1)!} \prod_{j=0}^{n} (x - x_j)\, f^{(n+1)}(\xi)$$

Ces résultats ainsi que l'expression de $p$ dans deux bases de polynômes sont montrés dans l'exercice 4.2.

**Interpolation d'Hermite :** Soient $x_0, \ldots, x_k$, $k+1$ réels distincts d'un intervalle $[a, b]$ ; on se donne $k+1$ entier naturels $\alpha_0, \ldots, \alpha_k$ et on pose $n = k + \alpha_0 + \ldots + \alpha_k$. Si $f$ est une fonction définie sur $[a, b]$ admettant des dérivées d'ordre $\alpha_i$ aux points $x_i$, *il existe un unique polynôme $p \in \mathbb{P}_n$ tel que $p^{(j)}(x_i) = f^{(j)}(x_i)$ pour $i = 0, \ldots, n$ et $j = 0, \ldots, \alpha_j$. Si $f \in \mathcal{C}^{n+1}([a, b])$ et $x \in [a, b]$, alors il existe $\xi$ appartenant au plus petit intervalle ouvert contenant $x$ et les $x_i$ tel que*

$$f(x) - p(x) = \frac{1}{(n+1)!} \Pi_n(x)\, f^{(n+1)}(\xi), \text{ où } \Pi_n(x) = \prod_{i=0}^{k} (x - x_i)^{\alpha_i + 1}$$

**Splines cubiques de classe $\mathcal{C}^2$ :** Soient $a = x_0 < x_2 < \ldots < x_n = b$, $n+1$ réels distincts. Pour tout ensemble de $n+3$ données $y'_0, y_0, \ldots, y_n, y'_n$, *il existe une unique fonction $S$ de $\mathbb{P}_3^2 = \left\{ \phi \in \mathcal{C}^2([a, b]) : \phi_{[x_i, x_{i+1}]} \in \mathbb{P}_3, 0 \leqslant i \leqslant n-1 \right\}$ vérifiant :*

$$S(x_i) = y_i,\ 0 \leqslant i \leqslant n,\ S'(x_0) = y'_0,\ S'(x_n) = y'_n$$

Si $y_i = f(x_i)$ pour $i = 0, \ldots, n$, et $y'_0 = f'(x_0)$, $y'_n = f'(x_n)$ où $f$ est une fonction de $\mathcal{C}^4([a,b])$, alors *pour tout x de* $[a,b]$, $|S(x) - f(x)| \leqslant \max_{t\in[a,b]} \left|f^{(4)}(t)\right| \dfrac{5h^4}{384}$ *où* $h = \max(x_{i+1} - x_i)$.

`Matlab` propose un certain nombre de méthodes pour réaliser l'interpolation de données. Étant donnés deux tableaux $x$ et $y$, abscisses et ordonnées, l'instruction `interp1` détermine l'interpolé aux points du tableau $x$ par plusieurs méthodes. L'aide en ligne précise les choix de `method`. On pourra aussi utiliser l'instruction `yy = spline(x,y,xx)` basée sur une spline cubique not a knot proposée par [7] et décrite dans l'exercice 4.4. La fonction `interp2` permet de réaliser des interpolations en dim 2.

## ÉNONCÉS DES EXERCICES

Dans ce chapitre, après un exercice de changement de base qui permet de retrouver les matrices, nous étudions l'interpolation de Lagrange et deux bases de polynômes sont proposées : Lagrange et Newton. Ensuite, nous étudions l'erreur d'approximation. Le phénomène de Runge permet de visualiser la différence entre interpolation (on impose de passer par un certain nombre de points) et approximation (on cherche une approche globale). Dans la partie suivante, nous proposons une méthode numérique pour une dérivation approchée. Nous étudions ensuite la construction des splines cubiques avec une étude numérique de l'erreur.

### 4.1 Changement de base

Nous étudions trois bases de $\mathbb{P}_3$. Sur $[0,1]$, nous définissons $B_0^3(t) = (1-t)^3$, $B_1^3(t) = 3t(1-t)^2$, $B_2^3(t) = 3t^2(1-t)$, $B_3^3(t) = t^3$.

**1.** Montrer que $\{B_i^3\}_{i=0,1,2,3}$ forme une base de $\mathbb{P}_3$. C'est la base de Bernstein de degré 3.

**2.** Montrer qu'on peut définir quatre fonctions polynômiale $H_i^3(t)$ interpolant les données d'Hermite au bords :

$$
\begin{aligned}
&H_0^3(0) = 1,\ (H_0^3)'(0) = 0,\ (H_0^3)'(1) = 0,\ H_0^3(1) = 0,\\
&H_1^3(0) = 0,\ (H_1^3)'(0) = 1,\ (H_1^3)'(1) = 0,\ H_1^3(1) = 0,\\
&H_2^3(0) = 0,\ (H_2^3)'(0) = 0,\ (H_2^3)'(1) = 1,\ H_2^3(1) = 0,\\
&H_3^3(0) = 0,\ (H_3^3)'(0) = 0,\ (H_3^3)'(1) = 0,\ H_0^3(1) = 1.
\end{aligned}
$$

**3.** Montrer que $\{H_i^3\}_{i=0,1,2,3}$ forme une base de $\mathbb{P}_3$ (base d'Hermite).

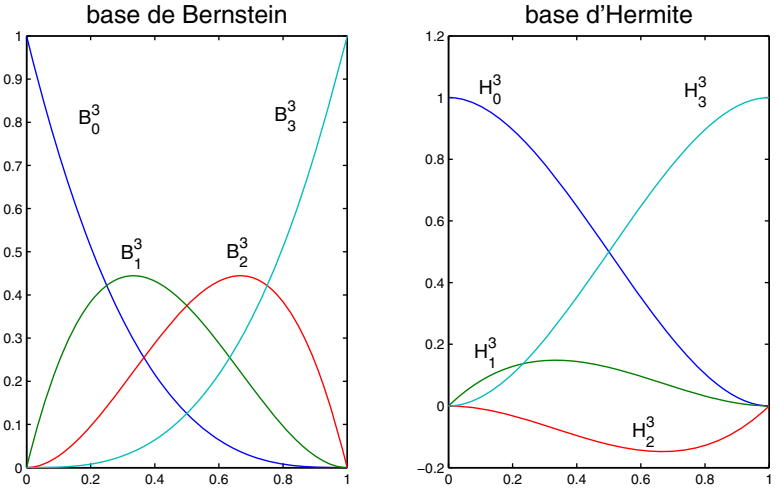
**4.** Déterminer l'interpolant d'Hermite des 4 données $f(0)$, $f'(0)$, $f'(1)$, $f(1)$.

**5.** Écrire la base $\{B_i^3\}_{i=0,1,2,3}$ dans la base canonique sous la forme : $\begin{pmatrix} B_0^3(t) \\ B_1^3 t) \\ B_2^3(t) \\ B_3^3(t) \end{pmatrix} = A \begin{pmatrix} 1 \\ t \\ t^2 \\ t^3 \end{pmatrix}$ où $A \in \mathbb{R}^{4\times 4}$. De même écrire la base $\{H_i^3\}_{i=0,1,2,3}$ dans la base canonique.

**6.** En déduire l'expression des polynômes $H_i^3$ dans la base de Bernstein.

**7.** Déterminer l'interpolant d'Hermite des quatre données $f(0)$, $f'(0)$, $f'(1)$, $f(1)$ dans la base de Bernstein.

**8.** À l'aide de `Matlab`, tracer les fonctions polynômiale $\{B_i^3\}_{i=0,1,2,3}$ puis $\{H_i^3\}_{i=0,1,2,3}$ sur $[0, 1]$.




## 4.2 Interpolation de Lagrange

Soit $a = x_1 < x_2 < \ldots < x_{n+1} = b$ une subdivision de l'intervalle $[a, b]$. On se donne $n + 1$ réels $y_i$.

**1.** **Existence et unicité**

Pour $i = 1, \ldots, n + 1$, on définit $\ell_i(x) = \prod_{\substack{j=1 \\ j\neq i}}^{n+1} \frac{x - x_j}{x_i - x_j}$. Montrer que $\{\ell_1, \ell_2, \ldots, \ell_{n+1}\}$ est une base de $\mathbb{P}_n = \mathbb{R}_n[X]$. On l'appelle base de Lagrange de $\mathbb{P}_n$.

**2.** Montrer qu'il existe un unique polynôme $p$ de degré inférieur ou égal à $n$ tel que $p(x_j) = y_j$ pour $j = 1$ à $n + 1$.

**3.** **Changement de base**

On note $p_k$ le polynôme d'interpolation des $k + 1$ premières données, i.e. $(x_j, y_j)$, $j = 1$ à $k + 1$.

(a) Montrer que $p_k(x) - p_{k-1}(x) = [y_{1,\ldots,}y_{k+1}](x - x_1)\cdots(x - x_k)$ où $[y_1, \ldots, y_{k+1}]$ est le coefficient de $x^k$ dans $p_k(x)$.

(b) En définissant $[y_j] = y_j$ , $j = 1$ à $n + 1$, montrer que $\forall k \geqslant 1$,

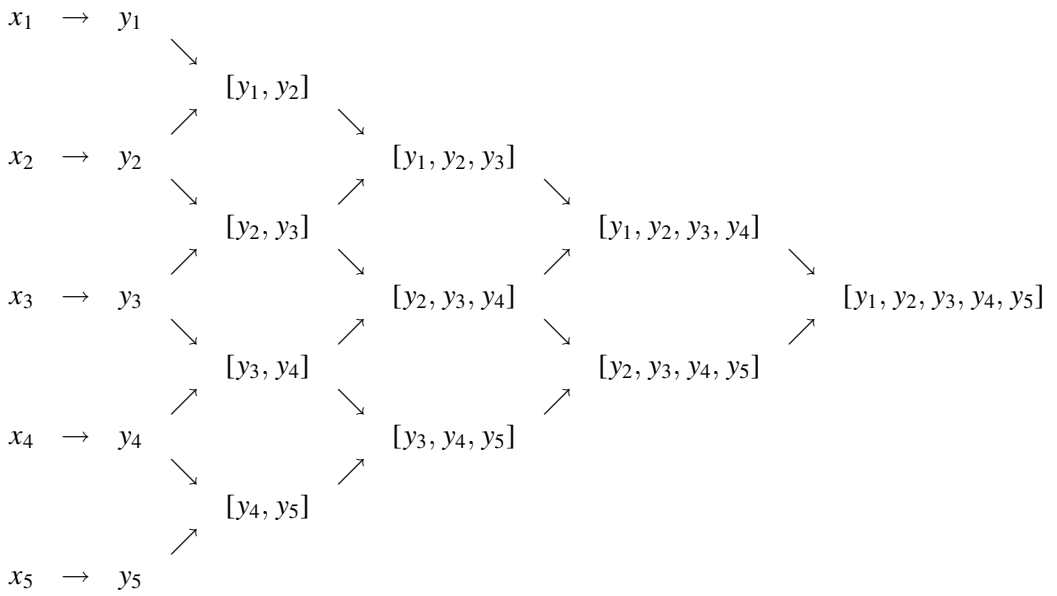
$$[y_1, \ldots, y_{k+1}] = \frac{[y_2, ..., y_{k+1}] - [y_1, \ldots, y_k]}{x_{k+1} - x_1} \ .$$

(c) Déduire de (a) que

$$p_n(x) = [y_1] + \sum_{k=1}^{n} [y_1, ..., y_{k+1}](x - x_1)...(x - x_k)$$

$\{1, x - x_1, (x - x_1)(x - x_2), \ldots, (x - x_1)\ldots(x - x_n)\}$ s'appelle la base de Newton. Les expressions $[y_1, ..., y_{k+1}]$ sont les différences divisées .

(d) Exemple $x_j = -3 + j$ , $j = 1$ à 5, $y_j = (-1)^j$ .
Dresser le tableau suivant :




## 4. Erreur d'approximation

On suppose que les données résultent de l'échantillonnage d'une fonction $f \in \mathcal{C}^{n+1}([a, b])$, $f(x_j) = y_j$ , $j = 1$ à $n+1$. On désigne l'erreur par $e(x) = f(x) - p(x)$. Naturellement $e(x_i) = 0$. Si $x \neq x_i$ $i = 1, \ldots, n + 1$, on définit la fonction $g$ par

$$g(t) = f(t) - p(t) - e(x) \prod_{j=1}^{n+1} \frac{(t - x_j)}{(x - x_j)}.$$

À noter que si $f$ est définie en dehors de $[a, b]$ et régulière, on peut aussi choisir $x$ en dehors de cet intervalle.

(a) Montrer que $g(x_1) = g(x_2) = \cdots = g(x_{n+1}) = 0$ et $g(x) = 0$.
En déduire qu'il existe $x_1^1 < x_2^1 < \cdots < x_{n+1}^1$ tel que

$$g'(x_1^1) = g'(x_2^1) = \cdots = g'(x_{n+1}^1) = 0$$

(b) Montrer qu'il existe un point $x_1^{n+1} = \xi$ tel que

$$g^{(n+1)}(\xi) = 0$$

(c) En déduire qu'il existe $\xi$ appartenant à l'intervalle ouvert contenant $x$ et les $x_i$ tel que

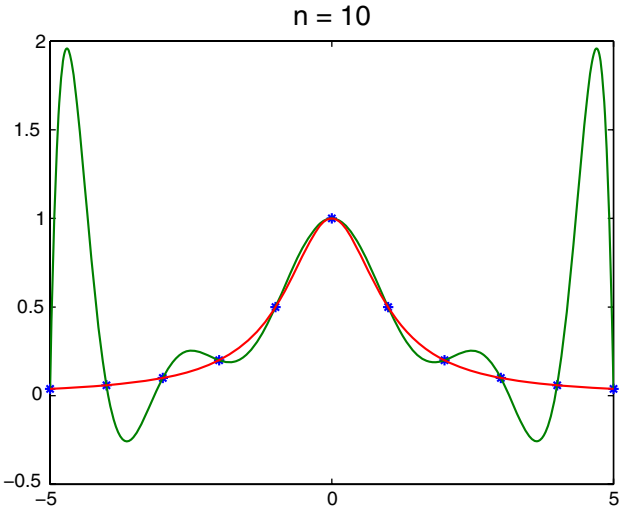
$$e(x) = \frac{1}{(n+1)!} \prod_{j=1}^{n+1} (x - x_j)\, f^{(n+1)}(\xi) \tag{4.1}$$

## 5. Le phénomène de Runge

On échantillonne $f(x) = \dfrac{1}{1+x^2}$ en $n+1$ points équidistants sur $[-5, 5]$ ; $x_j = -5 + 10\,\dfrac{j}{n}$ pour $j = 0, \ldots, n$.

Écrire un programme à l'aide de `Matlab` pour compléter un tableau *tabl* contenant une suite de valeurs de $n$, les valeurs de $f$, $p$ et $e$ au point $x_{n-\frac{1}{2}} = 5 - 5/n$. Pour $n$ donné, on pourra aussi tracer le graphe de $f$ et celui de $p$. Pour obtenir l'interpolant, utiliser successivement les instructions `coeff=polyfit(x,y,n)` qui donne les coefficients du polynôme de degré $n$ approchant les $n+1$ points $(x_i, y_i)$ au sens de moindres carrés, donc le polynôme de Lagrange, $p_n$, puis `polyval(coeff,t)` pour calculer la valeur de $p_n$ en un ou des points $t$. Test avec les valeurs de $n$ contenues dans le tableau $tabn = 2 : 2 : 20$.

```
>> tabl =
    2.0000    0.1379    0.7596    0.6217
    4.0000    0.0664   -0.3568    0.4232
    6.0000    0.0545    0.6079    0.5534
    8.0000    0.0497   -0.8310    0.8807
   10.0000    0.0471    1.5787    1.5317
   12.0000    0.0454   -2.7550    2.8004
   14.0000    0.0443    5.3327    5.2884
   16.0000    0.0435  -10.1739   10.2174
   18.0000    0.0429   20.1237   20.0808
   20.0000    0.0424  -39.9524   39.9949

       n       f(x)       p(x)     erreur
```




On notera ainsi que l'interpolant qui passe par les points $(x_i, f(x_i))_{i=1,\dots,n}$ ne réalise pas forcément une bonne approximation. Néanmoins, si on peut choisir les abscisses $x_i$, on peut avoir intérêt à minimiser la norme infinie du polynôme $\prod_{j=1}^{n+1}(x - x_j)$ sur $[a, b]$ qui apparaît dans (4.1) ; cette minimisation est obtenue aux points de Tchebychev.

## 4.3 Dérivation approchée

On considère une fonction définie sur $[a, b]$ dont on souhaite approcher la dérivée. Pour cela on utilise l'interpolation de Lagrange. Soit $X = (x_1, \dots, x_{N+1})^T$ un vecteur d'abscisses et $y = f(x)$, c'est-à-dire $Y = \big(f(x_1), \dots, f(x_{N+1})\big)^T$ un vecteur de valeurs associées (valeurs échantillonnées). On interpole la fonction au moyen d'un polynôme de Lagrange de degré $N$

$$P(t) = \sum_{j=1}^{N+1} y_j \, \ell_j(t) \;\text{ avec }\; \ell_j(t) = \prod_{\substack{k=1 \\ k\neq j}}^{N+1} \frac{t - x_k}{x_j - x_k}$$

**1.** Montrer que le vecteur des dérivées de $P$ aux points $x_i$, noté $D = \big(P'(x_1), \dots, P'(x_{N+1})\big)^T$ s'écrit sous la forme

$$D = MY \;,\; M \in \mathbb{R}^{(N+1)\times(N+1)}$$

où $m_{ij}$ est donné par

$$m_{ij} = \ell_j'(x_i) = \begin{cases} \dfrac{c_i}{c_j(x_i - x_j)} & \text{si } i \neq j \\ \displaystyle\sum_{\substack{k=1 \\ k\neq j}}^{N+1} \frac{1}{(x_i - x_k)} & \text{si } i = j \end{cases}$$

avec $c_i = \prod_{\substack{k=1 \\ k\neq i}}^{N+1}(x_i - x_k)$. Ce vecteur donnera l'approximation des dérivées de $f$ aux point $x_i$

**2.** On suppose que $[a, b] = [-2, 2]$. Écrire un programme `DA1.m` prenant $N$ comme argument, et produisant les points de Tchebychev

$$x_i = \frac{a+b}{2} + \frac{a-b}{2}\cos\left(\pi\left(\frac{i-1}{N}\right)\right),\ 1 \leqslant i \leqslant N+1$$

ainsi que les valeurs de $f$ correspondantes, sous forme de deux vecteurs colonnes $X$ et $Y$. Test avec $N = 5$, et $f = \arctan(x)$.

```
>> DA1
x =

   -2.0000
   -1.6180
   -0.6180
    0.6180
    1.6180
    2.0000

y =
   -1.1071
   -1.0172
   -0.5536
    0.5536
    1.0172
    1.1071
```

**3.** On construit la matrice $Q$ de $\mathbb{R}^{(N+1)\times(N+1)}$ définie par $Q_{ij} = \begin{cases} 1 \text{ si } i = j \\ \frac{1}{x_i - x_j} \text{ si } i \neq j \end{cases}$.

Compléter le premier programme en `DA2.m` affichant $Q$ où les points $(x_i)$ sont calculés par le premier programme. Test avec les mêmes données que précédemment.

```
>> DA2
Q =
    1.0000   -2.6180   -0.7236   -0.3820   -0.2764   -0.2500
    2.6180    1.0000   -1.0000   -0.4472   -0.3090   -0.2764
    0.7236    1.0000    1.0000   -0.8090   -0.4472   -0.3820
    0.3820    0.4472    0.8090    1.0000   -1.0000   -0.7236
    0.2764    0.3090    0.4472    1.0000    1.0000   -2.6180
    0.2500    0.2764    0.3820    0.7236    2.6180    1.0000
```

**4.** On remarque que $a_i = \prod_{\substack{k=1 \\ k\neq i}}^{N+1} \frac{1}{(x_i - x_k)} = \prod_{j=1}^{N+1} Q_{i,j}$.

On remarque d'autre part que $V_i = \sum_{j=1}^{N+1} Q_{i,j} = \sum_{\substack{j=1 \\ j\neq i}}^{N+1} \frac{1}{x_i - x_j} + 1$.

Écrire un programme, `DA3.m`, qui calcule et affiche les matrices diagonales. Utiliser `sum` et `prod`. Mêmes données test.

$$A = \begin{pmatrix} a_1 & & 0 \\ & \ddots & \\ 0 & & a_{N+1} \end{pmatrix} \quad \text{et} \quad V = \begin{pmatrix} V_1 - 2 & & 0 \\ & \ddots & \\ 0 & & V_{N+1} - 2 \end{pmatrix}$$

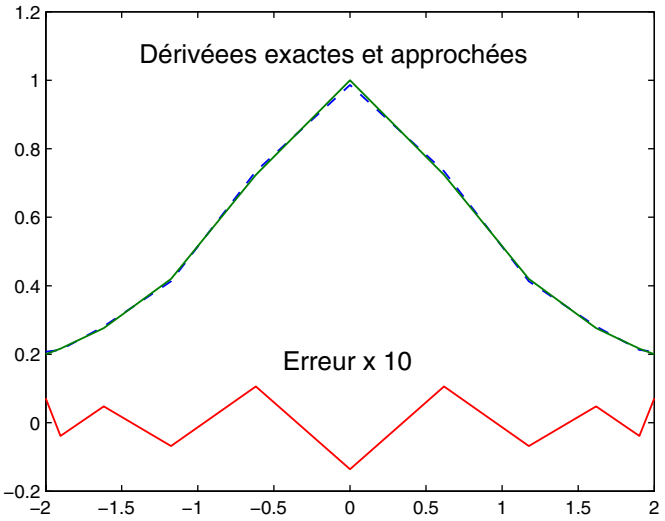
```
>> DA3
A =
   -0.0500         0         0         0         0         0
         0    0.1000         0         0         0         0
         0         0   -0.1000         0         0         0
         0         0         0    0.1000         0         0
         0         0         0         0   -0.1000         0
         0         0         0         0         0    0.0500

V =
   -5.2500         0         0         0         0         0
         0   -0.4146         0         0         0         0
         0         0   -0.9146         0         0         0
         0         0         0   -1.0854         0         0
         0         0         0         0   -1.5854         0
         0         0         0         0         0    3.2500
```

**5.** La matrice $M$ est donnée par $M = A^{-1}QA + V$ . En regroupant les programmes précédents, écrire un programme `DA4.m` qui à partir de $N$ produit le vecteur des dérivées approchées $D$. Test avec $N = 5$.

```
>> DA4
D =
    0.3177
    0.2140
    0.7738
    0.7738
    0.2140
    0.3177
```

**6.** Afficher sur une même figure la dérivée exacte, la dérivée approchée, et l'erreur entre les deux. Sauvegarder dans `DA5.m`.

Test avec $N = 10$, et $f = \arctan(x)$. On rappelle que $\arctan'(x) = \frac{1}{1+x^2}$ .




*Attention, l'erreur a été multipliée par 10*

**7.** Reprendre le programme avec cette fois-ci un échantillonnage en des points équidistants. Sauvegarde dans `DA6.m`.

**8.** Trouver au moins deux fonctions indépendantes pour lesquelles cette méthode de dérivation approchée donne la dérivée exacte.

**9.** La matrice $M$ n'est pas inversible. Donner dans le programme `DA7.m` le $Y_0$ non nul tel que

$$\|Y_0\|_2 = 1 \text{ et } \|MY_0\|_2 \simeq 0.$$

## 4.4 Splines cubiques

Au lieu de construire un polynôme interpolant, nous allons construire un interpolant qui sera polynômial par morceaux. La construction se fait en plusieurs étapes mais l'objectif est le même que pour l'interpolation de Lagrange : trouver une fonction $s$ telle que $s(x_i) = y_i$ (en reprenant les notations précédentes).

**1.** On souhaite construire une fonction polynômiale sur $[a, b]$ interpolant les données suivantes $y_a, y_b, y'_a$ et $y'_b$. Montrer qu'il existe un unique polynôme $P$ de degré 3 réalisant l'interpolation. On l'écrira dans la base de Bernstein : $B_0^3(t) = (1-t)^3$, $B_1^3(t) = 3t(1-t)^2$, $B_2^3(t) = 3t^2(1-t)$, $B_3^3(t) = t^3$ où $t = \dfrac{x-a}{b-a}$ en déterminant $p(x) = \displaystyle\sum_{i=0}^{3} \alpha_k B_k^3(t)$. On calculera ensuite $\dfrac{dB_k^3}{dt}$ et $\dfrac{dp}{dx}$.

**2.** Pour la suite, calculer les dérivées suivantes : $\dfrac{d^2 B_k^3}{dt^2}$ puis $\dfrac{d^2 p}{dx^2}$.

**3.** Pour une subdivision de $[a,b]$ : $x_0 = a < x_1 < ... < x_n = b$, on considère l'espace

$$\mathbb{P}_3^1 = \left\{ S \in \mathcal{C}^1(a,b) : S\,|_{[\,x_i, x_{i+1}\,]} \in \mathbb{P}_3,\ 0 \leqslant i \leqslant n-1 \right\}$$

Montrer que pour tout système de $2n+2$ données, il existe une unique fonction $S$ de $\mathbb{P}_3^1$ vérifiant

$$S(x_i) = y_i, S'(x_i) = y_i',\ 0 \leqslant i \leqslant n$$

$S$ est la spline cubique de classe $\mathcal{C}^1$ interpolante.

**4.** On considère $\mathbb{P}_3^2 = \left\{ S \in \mathcal{C}^2([a,b]) : S\,|_{[x_i, x_{i+1}]} \in \mathbb{P}_3, 0 \leqslant i \leqslant n-1 \right\}$, nous allons montrer le théorème suivant (démonstration dans le cas où $x_{i+1} - x_i$ est constant).

Pour tout ensemble de $n+3$ données $y_0', y_0, y_1, y_2, \ldots, y_{n-1}, y_n, y_n'$, il existe une unique fonction $S$ de $\mathbb{P}_3^2$ vérifiant :

$$S(x_i) = y_i,\ 0 \leqslant i \leqslant n,\ S'(x_0) = y_0',\ S'(x_n) = y_n'$$

(a) Dès que les pentes $y_i'$ en les $x_i$ seront connues, on sera ramené à la question précédente. Montrer que le raccord $\mathcal{C}^2$ au point $x_i$ donne une équation liant $y_{i-1}', y_i',\ y_{i+1}'$ et $y_{i-1},\ y_i,\ y_{i+1}$.

(b) En déduire que le vecteur inconnu $Y' = (y_1', \ldots, y_{n-1}')^T$ est solution d'un système tridiagonal $AY' = b$, $A \in \mathbb{R}^{(n-1)\times(n-1)}$ , à diagonale dominante stricte i.e

$$\forall i \in \{1, \ldots, n-1\},\ a_{ii} > \sum_{j=1,\ j\neq i}^{n-1} |a_{ij}|$$

(c) Montrer la proposition suivante (disques de Gershgorin) :
Théorème : *Si $M = (m_{ij}) \in \mathbb{C}^{k\times k}$ et $\lambda \in \mathbb{C}$ est une valeur propre de $M$ alors*

$$\lambda \in \bigcup_{i=1}^{k} D_i,\ \textit{où}\ D_i = \left\{ z \in \mathbb{C} : |z - m_{ii}| \leqslant \sum_{j=1,\ j\neq i}^{k} |m_{ij}| \right\} \tag{4.2}$$

(d) En déduire que 0 n'est pas valeur propre de $A$ donc que le système $AY' = b$ admet une solution unique.

On montre que si $f \in \mathcal{C}^4[a,b]$, que $M_4 = \sup_{x\in[a,b]} |f^{(4)}(x)|$ et $h = \max_{i=1,\ldots,n} |x_{i+1} - x_i|$, alors

$$\forall x \in [a,b],\ \begin{cases} |f(x) - S(x)| & \leqslant M_4 \dfrac{5h^4}{384} \\ |f'(x) - S'(x)| & \leqslant M_4 \dfrac{h^3}{24} \\ |f''(x) - S''(x)| & \leqslant M_4 \dfrac{3h^2}{8} \end{cases}$$

### 5. Splines not a knot, (pas constant)

On suppose que $y_0'$ et $y_n'$ sont inconnus et on impose, en plus des $n-1$ conditions de raccord $\mathcal{C}^2$, les 2 conditions supplémentaires de raccord $\mathcal{C}^3$ en $x_1$ et $x_{n-1}$. Montrer que le système à $n+1$ inconnues peut s'écrire

$$\left\{\begin{array}{l} y_0' - y_2' = -\dfrac{2}{h}(y_0 - 2y_1 + y_2) \\ A1 \begin{pmatrix} y_1' \\ \vdots \\ \vdots \\ y_{n-1}' \end{pmatrix} = k \\ y_{n-2}' - y_n' = -\dfrac{2}{h}(y_{n-2} - 2y_{n-1} + y_n) \end{array}\right. \tag{4.3}$$

où

$$A1 = \begin{pmatrix} 4 & 2 & 0 & \dots & 0 \\ 1 & 4 & 1 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & 1 & 4 & 1 \\ 0 & \dots & 0 & 2 & 4 \end{pmatrix} \in \mathbb{R}^{(n-1)\times(n-1)},\ k = \frac{1}{h}\begin{pmatrix} 5y_2 - 4y_1 - y_0 \\ 3(y_2 - y_0) \\ \vdots \\ 3(y_n - y_{n-2}) \\ -5y_{n-2} + 4y_{n-1} + y_n \end{pmatrix} \in \mathbb{R}^{n-1}.$$

Ce système admet encore une solution unique puisque $A1$ reste à diagonale dominante stricte.

### 6. Programmation

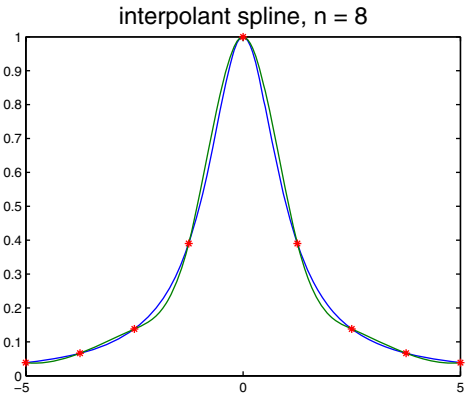
Étant donnés deux tableaux $x$ et $y$ de même dimension $n$, les $x_i$ étant distincts et un tableau $xabs$, l'instruction `yord = spline (x,y,xabs)` calcule automatiquement la spline not a knot aux points $xabs$.

(a) Construire un fichier `f.m`. Étant donné un entier $n$, construire un tableau $x$ avec un pas constant et déterminer le tableau $y = f(x)$. Calculer la spline, dessiner les graphes de $f$ et de la spline.

Exemples : $f1(x) = e^x$ sur $[0,1]$, $\quad f2(x) = \dfrac{1}{1+x^2}$ sur $[-5,5]$

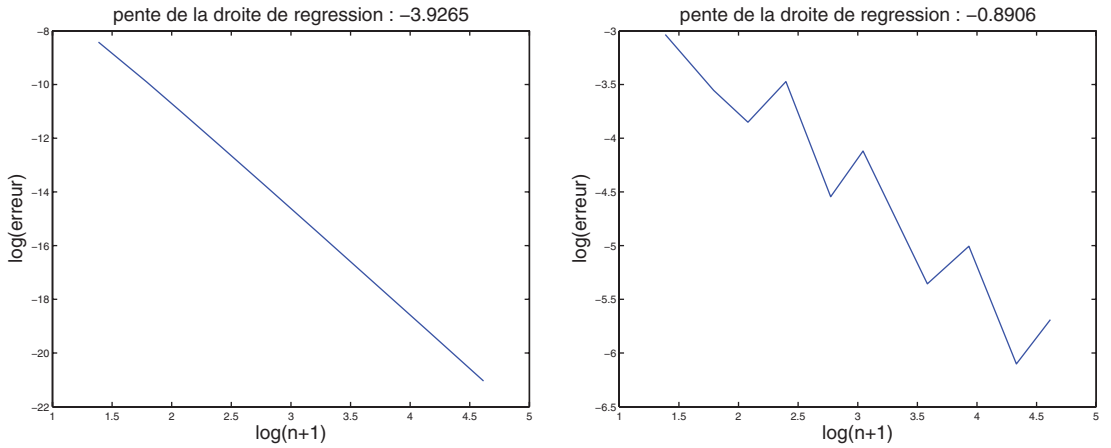
$$f3(x) = \left\{\begin{array}{ll} x & \text{si } x < 0 \\ x/2 & \text{si } x \geqslant 0 \end{array}\right. \quad \text{sur } [-2,2].$$

En augmentant $n$, on peut constater que le phénomène de Runge n'apparaît pas.




(b) Dans les cas $f1$ et $f2$, étudier l'erreur quand $n$ varie. On partira d'un tableau $tn$ et on calculera le tableau $taberreur$ puis on tracera $\log(taberreur)$ en fonction de $\log(tn+1)$.

Ce tracé permet de visualiser numériquement que $err \simeq c/N^{\alpha}$ en déterminant $\alpha$, pente de la droite.




*erreurs pour les fonctions $f1$ et $f3$*

Dans le premier cas, pour une fonction échantillonnée de classe $\mathcal{C}^{\infty}$, on retrouve une erreur en $h^4$ alors que dans le second cas, la fonction échantillonnée est seulement de classe $\mathcal{C}^0$ ; on peut montrer que dans ce cas, l'erreur se majore par $O(h)$.

## DU MAL À DÉMARRER ?

### 4.1 Changement de base

**1.** Montrer qu'il s'agit d'un système libre.

**2.** 0 ou 1 sont racines éventuellement multiples.

**4.** On peut utiliser la base précédente.

### 4.2 Interpolation de Lagrange

**1.** Montrer que ces fonctions forment un système libre.

**2.** Utiliser la base précédente.

**4.** Le théorème de Rolle peut être utile.

### 4.3 Dérivation approchée

**2.** On peut commencer par construire la matrice $Q1$ telle que $Q1_{ij} = x_i - x_j$, puis lui ajouter la matrice identité.

### 4.4 Splines cubiques

**1.** On peut faire un calcul direct ou s'inspirer de l'exercice 4.1.

**4.** (a) Pour déterminer les équations liant les dérivées premières, écrire que les dérivées secondes à gauche et à droite en les $x_i$ coïncident.

(c) Écrire $\lambda x_i = \sum_{j=1}^{k} m_{ij} x_j$ en choisissant un indice $i$ tel que $|x_i| = \max |x_j|$.

## CORRIGÉS DES EXERCICES

### 4.1 Changement de base

**1.** Puisque $\{B_i^3\}_{i=0,1,2,3}$ a 4 éléments et que $\mathbb{P}_3$ est de dimension 4, il suffit de montrer que le système est libre pour conclure qu'il forme une base. Si $\sum_{i=0}^{3} \lambda_i B_i^3 = O$, alors en $t = 0$, il vient $\lambda_0 = 0$. De même en $t = 1$, $\lambda_3 = 0$. En dérivant $\sum_{i=1}^{2} \lambda_i B_i^3 = O$ puis en prenant à nouveau $t = 0$ puis $t = 1$, nous trouvons $\lambda_1 = \lambda_2 = 0$.

**2.** $H_0^3$ est un polynôme de degré 3 tel que $H_0^3(0) = 1,\ (H_0^3)'(0) = 0,\ (H_0^3)'(1) = 0,\ H_0^3(1) = 0$. Donc 1 est racine double si bien que $H_0^3(t) = (1-t)^2(at+b)$. Les 2 conditions restantes permettent de déterminer $a$ et $b$ et finalement $H_0^3(t) = (1-t)^2(2t+1)$. De même $H_1^3(t) = t(1-t)^2$, $H_2^3(t) = t^2(t-1)$ et $H_3^3(t) = t^2(3-2t)$.

**3.** De même que dans la question 1, on montre que $\{H_i^3\}_{i=0,1,2,3}$ forme une base de $\mathbb{P}_3$.

**4.** L'interpolant d'Hermite des 4 données $f(0)$, $f'(0)$, $f'(1)$, $f(1)$ est donné par $p = f(0)H_0^3 + f'(0)H_1^3 + f'(1)H_2^3 + f(1)H_3^3$ puisqu'il est unique et que ce dernier répond à la question.

**5.** En développant les polynômes, on obtient $\begin{pmatrix} B_0^3(t) \\ B_1^3 t) \\ B_2^3(t) \\ B_3^3(t) \end{pmatrix} = A \begin{pmatrix} 1 \\ t \\ t^2 \\ t^3 \end{pmatrix}$ et $\begin{pmatrix} H_0^3(t) \\ H_1^3 t) \\ H_2^3(t) \\ H_3^3(t) \end{pmatrix} = B \begin{pmatrix} 1 \\ t \\ t^2 \\ t^3 \end{pmatrix}$ où $A = \begin{pmatrix} 1 & -3 & 3 & -1 \\ 0 & 3 & -6 & 3 \\ 0 & 0 & 3 & -3 \\ 0 & 0 & 0 & 1 \end{pmatrix}$ et $B = \begin{pmatrix} 1 & 0 & -3 & 2 \\ 0 & 1 & -2 & 1 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 3 & -2 \end{pmatrix}$.

**6.** Donc $\begin{pmatrix} H_0^3(t) \\ H_1^3 t) \\ H_2^3(t) \\ H_3^3(t) \end{pmatrix} = BA^{-1} \begin{pmatrix} B_0^3(t) \\ B_1^3 t) \\ B_2^3(t) \\ B_3^3(t) \end{pmatrix}$. $A$ est triangulaire et $A^{-1}$ peut se déterminer en commençant par la dernière ligne : $A^{-1} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1/3 & 2/3 & 1 \\ 0 & 0 & 1/3 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$ et $BA^{-1} = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 1/3 & 0 & 0 \\ 0 & 0 & -1/3 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix}$.

**7.** $p = f(0)H_0^3 + f'(0)H_1^3 + f'(1)H_2^3 + f(1)H_3^3$. En écrivant les $H_i^3$ dans la base de Bernstein, il vient $p = f(0)B_0^3 + (f(0) + f'(0)/3)B_1^3 + (f(1) - f'(1)/3)B_2^3 + f(1)B_3^3$.

**8.** `bases.m`

```
t=0:1/500:1;
B0=(1-t).^3; B1=3*t.*(1-t).^2;
B2=3*t.^2.*(1-t); B3=t.^3;
subplot(1,2,1)
plot(t,B0,t,B1,t,B2,t,B3)
title('base de Bernstein','Fontsize',18)
H0=(t-1).^2.*(1+2*t); H1=t.*(1-t).^2;
H2=(t-1).*t.^2; H3=t.^2.*(3-2*t);
subplot(1,2,2)
plot(t,H0,t,H1,t,H2,t,H3)
title('base d''Hermite','Fontsize',18)
```

## 4.2 Interpolation de Lagrange

**1.** Le système $\{\ell_1, \ldots, \ell_{n+1}\}$ contient $n+1$ éléments dans l'espace des polynômes $\mathbb{P}_n = \mathbb{R}_n[X]$ qui est de dimension $n+1$. Pour montrer que ce système forme une base, il suffit de montrer qu'il est libre. Nous avons $\ell_k(x_i) = \delta_{ik} = \begin{cases} 1 & \text{si } i = k \\ 0 & \text{si } i \neq k \end{cases}$. Donc si $\sum_{k=1}^{n+1} \lambda_k \ell_k = 0$, en prenant la valeur

$x_i$, il vient $\lambda_i = 0$ pour $i = 1, \ldots, n+1$. Maintenant il est aussi clair que $y_i = \sum_{k=1}^{n+1} y_k \ell_k(x_i)$ donc $p$ défini par $p(x) = \sum_{k=1}^{n+1} y_k \ell_k(x)$ est de degré $n$ et réalise l'interpolation. L'unicité est obtenue puisque le système $\{\ell_1, \ldots, \ell_{n+1}\}$ est une base.

**2.** $p_k - p_{k-1}$ est un polynôme de degré inférieur ou égal à $k$ qui s'annule en chacun des $x_i$ pour $i = 1, \ldots, k$. On a donc $p_k(x) - p_{k-1}(x) = a_k \prod_{i=1}^{k}(x - x_i)$. Le coefficient $a_k$ est celui de $x^k$. Or $p_{k-1}$ est de degré strictement inférieur à $k$. Donc $a_k$ ne peut être que le coefficient de $x^k$ dans $p_k$. On note $a_k = [y_1, \ldots, y_{k+1}]$. Si bien que

$$p_k(x) - p_{k-1}(x) = [y_1, \ldots, y_{k+1}] \prod_{i=1}^{k}(x - x_i) \tag{4.4}$$

Soit $q_{k-1}$ le polynôme interpolant les points $(x_i, y_i)$, $i = 2, \ldots, k+1$ ; $q_{k-1}$ est un polynôme de degré inférieur ou égal à $k-1$ et le coefficient de $x^{k-1}$ est noté $[y_2, \ldots, y_{k+1}]$. Par ailleurs, de même que précédemment,

$$p_k(x) - q_{k-1}(x) = [y_1, \ldots, y_{k+1}] \prod_{i=2}^{k+1}(x - x_i). \tag{4.5}$$

La différence $q_{k-1} - p_{k-1}$ est donc un polynôme de degré inférieur ou égal à $k-1$ et le coefficient de $x^{k-1}$ est $[y_2, \ldots, y_{k+1}] - [y_1, \ldots, y_k]$. Par ailleurs en effectuant la différence entre (4.4) et (4.5), on obtient

$$\begin{aligned} q_{k-1}(x) - p_{k-1}(x) &= [y_1, \ldots, y_{k+1}] \prod_{i=2}^{k}(x - x_i)\big[(x - x_{k+1}) - (x - x_1)\big] \\ &= [y_1, \ldots, y_{k+1}] \prod_{i=2}^{k}(x - x_i)(x_1 - x_{k+1}). \end{aligned}$$

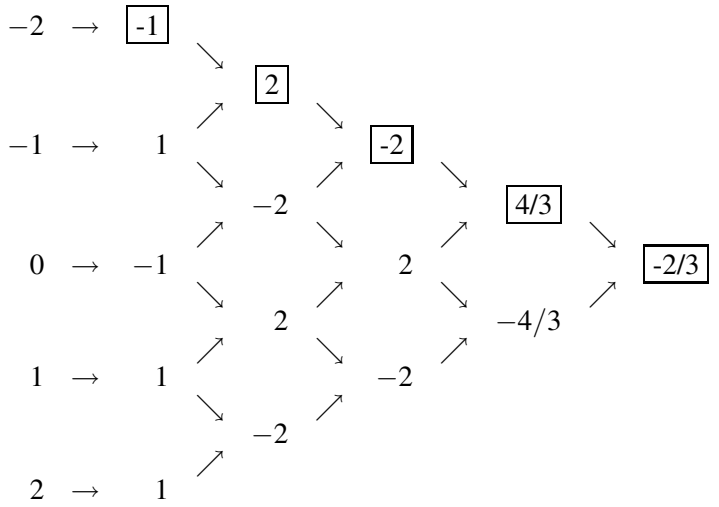
En comparant les coefficients de $x^{k-1}$ dans les 2 cas, il vient

$$[y_2, \ldots, y_{k+1}] - [y_1, \ldots, y_k] = [y_1, \ldots, y_{k+1}](x_1 - x_{k+1}).$$

Par récurrence, à partir de (4.4), on obtient

$$p_n(x) = [y_1] + \sum_{k=1}^{n} [y_1, ..., y_{k+1}](x - x_1)...(x - x_k).$$

Dans le cas particulier proposé, $x_j = -3 + j$ , $j = 1$ à 5, $y_j = (-1)^j$, on obtient alors le tableau des différences divisées




Les coefficients dans la base de Newton sont encadrés et le polynôme de Lagrange s'écrit : $p(x) = -1 + 2(x+2) - 2(x+2)(x+1) + 4/3(x+2)(x+1)x - 2/3(x+2)(x+1)x(x-1)$.

**3.** Il reste à évaluer l'erreur d'approximation. $x$ étant un réel situé dans le domaine où la fonction $f$ est définie avec $x \neq x_i$. Notons $Q(t) = \prod_{j=1}^{n+1} \frac{(t - x_j)}{(x - x_j)}$.

Sachant que $g(t) = f(t) - p(t) - e(x)Q(t)$ et $f(x_i) = p(x_i)$, $i = 1, \ldots, n+1$, on obtient $g(x_i) = 0$ et puisque $e(x) = f(x) - p(x)$ et $Q(x) = 1$, il vient $g(x) = 0$. Si bien que nous avons $k+2$ points distincts où la fonction $g$ s'annule. Dès que $f$ est dérivable, $g$ l'est aussi puisque $p$ et $Q$ sont des polynômes. Le théorème de Rolle nous assure alors qu'entre 2 points successifs où $g$ s'annule, il existe un point strictement compris entre les 2 points précédents où $g'$ s'annule Ainsi il existe une suite strictement croissante $x_i^1$, $i = 1, \ldots, n+1$ telle que $g'(x_1^1) = g(x_2^1) = \cdots = g(x_{n+1}^1) = 0$. À partir de là, on peut construire une nouvelle suite strictement croissante $x_i^2$, $i = 1, \ldots, n$ où $g''$ s'annule dès que $f$ est deux fois dérivable et, par récurrence, il existe un point $x_1^{n+1} = \xi$ tel que $g^{(n+1)}(\xi) = 0$ dès que $f$ est $n+1$ fois dérivable. En reprenant la définition de $g$ et en dérivant $n+1$ fois par rapport à $t$, sachant que $p^{(n+1)} = 0$ puisque $p$ est de degré inférieur ou égal à $n$, et que $Q^{(n+1)}(t) = \prod_{j=1}^{n+1} \frac{1}{(x - x_j)} \times (n+1)!$, on obtient $e(x) = \frac{1}{(n+1)!} \prod_{j=1}^{n+1} (x - x_j)\, f^{(n+1)}(\xi)$. Cette formule reste vraie quand $x$ coïncide avec l'un des $x_i$ initiaux car dans ce cas $e(x_i) = 0$.

**4.** lag1.m

```
clear
tabn=[2,4,6,8,10,12,14,16,18,20];
a=-5;b=5;
for i=1:length(tabn)
    n=tabn(i);
    h=(b-a)/n;
    x=a:h:b;
    y=1./(1+x.*x);
    coeff=polyfit(x,y,n);
    xnm=b-h/2;
    tabl(i,1)=n;
    tabl(i,2)=1/(1+xnm*xnm);
    tabl(i,3)=polyval(coeff,xnm);
    tabl(i,4)=abs(tabl(i,2)-tabl(i,3));
    t=a:0.01:b;
    pn=polyval(coeff,t);
    %plot(x,y,'*',t,pn,t,1./(1+t.*t));
    %n
    %title(['n = ',int2str(n)],'Fontsize',16)
    %pause
end;
tabl
disp('␣␣␣␣␣␣␣n␣␣␣␣␣␣␣f(x)␣␣␣␣␣␣␣p(x)␣␣␣␣␣erreur')
```

### Ce qu'il faut retenir de cet exercice

Pour faire varier $n$, on construit un tableau quelconque de valeurs croissantes de $n$, `tabn`, indicé par $i$ qui varie de 1 à la longueur du tableau.

On peut aussi programmer le calcul du polynôme de Lagrange dans une des bases proposées précédemment.

## 4.3 Dérivées approchées

**1.** Sachant que $P(t) = \sum_{j=1}^{N+1} y_j \, \ell_j(t)$ avec $\ell_j(t) = \prod_{\substack{k=1 \\ k\neq j}}^{N+1} \frac{t - x_k}{x_j - x_k}$, nous avons $P'(t) = \sum_{j=1}^{N+1} y_j \, \ell'_j(t)$.

Si $c_j = \prod_{\substack{k=1 \\ k\neq j}}^{N+1} (x_j - x_k)$, alors $\ell'_j(t) = \sum_{\substack{m=1 \\ m\neq j,}}^{N+1} \frac{\prod_{\substack{k=1 \\ k\neq j, k\neq m}}^{N+1} (t - x_k)}{\prod_{\substack{k=1 \\ k\neq j}}^{N+1} (x_j - x_k)} = \sum_{\substack{m=1 \\ m\neq j,}}^{N+1} \frac{1}{c_j} \prod_{\substack{k=1 \\ k\neq j, k\neq m}}^{N+1} (t - x_k)$,

$$\text{donc} \quad \ell_j'(x_i) = \begin{cases} \left( \prod_{\substack{k=1 \\ k\neq j, k\neq i}}^{N+1} (x_i - x_k) \right) / c_j = \dfrac{c_i}{c_j(x_i - x_j)} \text{ si } i \neq j, \\ \displaystyle\sum_{\substack{m=1 \\ m\neq i,}} \frac{c_i}{(x_i - x_m)c_i} = \sum_{\substack{m=1 \\ m\neq i,}} \frac{1}{x_i - x_m} \text{ si } i = j. \end{cases}$$

**2...5** `DA5.m`

```
clear
a=-2;b=2;
N=10;
%DA1
x=(a+b)/2+(a-b)*cos(pi*(0:N)/N)/2;
x=x';
y=f(x);
%DA2
Q=1./(eye(N+1)+x*ones(1,N+1)-ones(N+1,1)*x');
%DA3
A=diag(prod(Q,2));
V=diag(sum(Q,2)-2);
%DA4
M=A^(-1)*Q*A+V;
%DA5
D=M*y;
Dex=1./(1+x.*x);
plot(x,D,x,Dex,x,10*(D-Dex))
```

### Ce qu'il faut retenir de cet exercice

On a vu au chapitre 1 que les boucles retardaient l'exécution d'un programme. Ici la construction de la matrice $Q$ et des suivantes se fait donc sans boucle. Par ailleurs, l'aide en ligne est bien utile pour pouvoir apprendre à faire des sommes de tableaux suivant les lignes ou les colonnes.

**6.** Remplacer `x=(a+b)/2+(a-b)*cos(pi*(0 :N)/N)/2 ;` par `x=a :(b-a)/N :b ;` dans le programme précédent. L'erreur est plus important pour des points équidistants que pour les points de Tchebychev.

On peut remarquer que dans (4.1) qui donne l'erreur d'approximation pour la fonction, en choisissant les abscisses d'échantillonnage, on peut espérer minimiser $\max_{x\in[a,b]} \left| \prod_{j=1}^{n+1} (x - x_j) \right|$ ; les points de Tchebychev réalisent justement ce minimum. À défaut de le montrer, on peut tracer le graphe du polynôme $\Pi_n$ avec $\Pi_n(t) = \prod_{j=1}^{n+1} (t - x_j)$ dans les deux cas (Tchebychev et points équidistants) puis constater que pour $a = -2$ et $b = 2$, ce polynôme varie

de $-4$ à 4 indépendamment de $n$ pour les abscisses de Tchebychev et a des extrema tendant vers l'infini avec $n$ pour les points équidistants.

**7.** Dès que l'on interpole un polynôme $p$ de degré inférieur ou égal à $n$ en $n+1$ points, le polynôme de Lagrange coïncide avec $p$, par unicité et donc les dérivées coïncident aussi. Il suffit de prendre pour $f$ l'un des deux polynômes 1 ou $x$ dans le programme précédent en changeant aussi `Dex` pour avoir une erreur nulle.

**8.** Un polynôme constant a des dérivées exacte et approchée nulles. Si on veut $\|Y_0\|_2 = 1$, pour $N$ fixé, on partira de $f$ constante, $f = \dfrac{1}{\sqrt{N+1}}$ pour obtenir $MY_0 = 0$. À noter que le déterminant de $M$ est nul.

## 4.4 Splines cubiques

**1.** Il s'agit ici d'interpolation d'Hermite, fonction et dérivée. On remarque facilement que les $\{B_k^3\}$ forment une base de $\mathbb{P}_3$. Donc l'existence des coefficients $\alpha_k$ entraînera l'unicité. En dérivant par rapport à $t$, il vient :

$$\begin{aligned}
(B_0^3)'(t) &= -3(1-t)^2,\\
(B_1^3)'(t) &= 3(1-3t)(1-t),\\
(B_2^3)'(t) &= 3t(2-3t),\\
(B_3^3)'(t) &= 3t^2.
\end{aligned}$$

Notons que si $p(x) = \displaystyle\sum_{k=0}^{3} \alpha_k B_k^3(t)$, alors $\dfrac{dp}{dx} = \dfrac{dp}{dt}\dfrac{dt}{dx} = \dfrac{1}{b-a}\displaystyle\sum_{k=0}^{3} \alpha_k \dfrac{dB_k^3}{dt}$.

Les conditions aux bords s'écrivent alors :

$$\begin{aligned}
y_a &= p(a) = \sum_{k=0}^{3} \alpha_k B_k^3(0) = \alpha_0\\
y_a' &= \frac{dp}{dx}(a) = \frac{1}{b-a}\sum_{k=0}^{3} \alpha_k \frac{dB_k^3}{dt}(0) = 3\frac{-\alpha_0+\alpha_1}{b-a}\\
y_b &= p(b) = \sum_{k=0}^{3} \alpha_k B_k^3(1) = \alpha_3\\
y_b' &= \frac{dp}{dx}(b) = \frac{1}{b-a}\sum_{k=0}^{3} \alpha_k \frac{dB_k^3}{dt}(1) = 3\frac{-\alpha_2+\alpha_3}{b-a}
\end{aligned}$$

qui permettent d'obtenir

$$\begin{cases} \alpha_0 &= y_a \\ \alpha_1 &= y_a + \dfrac{b-a}{3} y'_a \\ \alpha_2 &= y_b - \dfrac{b-a}{3} y'_b \\ \alpha_3 &= y_b \end{cases} \tag{4.6}$$

**2.** Le calcul des dérivées secondes donne :
$(B_0^3)''(t) = 6(1-t)$, $(B_1^3)''(t) = 6(-2+3t)$, $(B_2^3)''(t) = 6(1-3t)$, $(B_3^3)''(t) = 6t$
et $\dfrac{d^2p}{dx^2}(x) = \dfrac{1}{(b-a)^2} \displaystyle\sum_{k=0}^{3} \alpha_k \frac{d^2 B_k^3}{dt^2}(t)$.

**3.** Étant donnée une subdivision $a = x_0 < x_1 < \ldots < x_n = b$, sur chaque intervalle $[x_i, x_{i+1}]$, on peut réaliser l'interpolation précédente. Puisque les valeurs de la fonction et de sa dérivée sont données en chaque $x_i$, l'interpolant polynômial par morceaux est de classe $\mathcal{C}^1$. L'unicité de l'élément de $\mathbb{P}_3^1$ résulte de l'unicité de la construction sur chaque intervalle.

**4.** **Condition nécessaire**
Nous supposons l'existence de $S \in \mathbb{P}_3^2$ réalisant l'interpolation et nous écrivons en chaque $x_i$, $i = 1, \ldots, n-1$ la continuité de la dérivée seconde de la fonction $S$ i.e $S''(x_{i+}) = S''(x_{i-})$. Nous notons $h = x_{i+1} - x_i$ et $y'_i$ les dérivées en chacun des $x_i$ ; elles sont inconnues sauf pour $i = 0$ et $i = n$.

Sur l'intervalle $[x_i, x_{i+1}]$, $S$ coïncide avec un polynôme de degré 3 du type de celui de la première question $S(x) = \displaystyle\sum_{k=0}^{3} \alpha_k^i B_k(t)$ avec $t = \dfrac{x - x_i}{h}$ et
$\alpha_0^i = y_i$, $\alpha_1^i = y_i + \dfrac{h}{3} y'_i$, $\alpha_2^i = y_{i+1} - \dfrac{h}{3} y'_{i+1}$, $\alpha_3^i = y_{i+1}$. Nous en déduisons

$$S''(x_{i+}) = \frac{1}{h^2} \sum_{k=0}^{3} \alpha_k^i \frac{d^2 B_k}{dt^2}(0) = \frac{6}{h^2} \left( y_{i+1} - y_i - \frac{h}{2}(y'_{i+1} + 2y_i) \right).$$

De même, sur l'intervalle $[x_{i-1}, x_i]$, $S(x) = \displaystyle\sum_{k=0}^{3} \alpha_k^{i-1} B_k(t)$ avec $t = \frac{x - x_{i-1}}{h}$ et
$\alpha_0^{i-1} = y_{i-1}$, $\alpha_1^{i-1} = y_{i-1} + \frac{h}{3} y'_{i-1}$, $\alpha_2^{i-1} = y_i - \frac{h}{3} y'_i$, $\alpha_3^{i-1} = y_i$ ; donc

$$S''(x_{i-}) = \frac{1}{h^2} \sum_{k=0}^{3} \alpha_k^{i-1} \frac{d^2 B_k}{dt^2}(1) = \frac{6}{h^2} \left( y_{i-1} + y_i - \frac{h}{2}(y'_{i-1} + 2y_i) \right).$$

En écrivant $S''(x_{i+}) = S''(x_{i-})$, il vient :

$$y'_{i-1} + 4y'_i + y'_{i+1} = \frac{3}{h}(y_{i+1} - y_{i-1}),\ i = 1, \ldots, n-1,$$

soit le système $AY' = b$ où $Y' = (y'_1, \ldots, y'_{n-1})^T$ est l'inconnue et

$$A = \begin{pmatrix} 4 & 1 & 0 & \ldots & \ldots & 0 \\ 1 & 4 & 1 & 0 & \ldots & \vdots \\ 0 & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & 0 & 1 & 4 & 1 \\ 0 & \ldots & \ldots & 0 & 1 & 4 \end{pmatrix},\ b = \frac{3}{h}\begin{pmatrix} y_2 - y_0 \\ y_3 - y_1 \\ \vdots \\ \vdots \\ y_{n-1} - y_{n-3} \\ y_n - y_{n-2} \end{pmatrix} - \begin{pmatrix} y'_0 \\ 0 \\ \vdots \\ \vdots \\ 0 \\ y'_n \end{pmatrix}.$$

$A$ est tridiagonale et à diagonale dominante stricte.

Nous laissons le lecteur montrer que la condition $AY' = b$ est suffisante.

Il reste à montrer que le système admet une solution unique. Pour cela nous allons montrer la proposition plus générale qui localise les valeurs propres d'une matrice carrée $M \in \mathbb{C}^{k\times k}$.

Soit $\lambda$ une valeur propre de $M = (m_{ij})$ et $X \neq 0$ un vecteur propre correspondant. Appelons $i$ un indice tel que $|x_i| = \max\limits_{j=1,\ldots,k}\{|x_j|\}$. $x_i \neq 0$ car $X \neq 0$. Nous avons $\lambda x_i = \sum\limits_{i=1}^{k} m_{ij}x_j$ donc

$$|\lambda - m_{ii}| \times |x_i| = \left|\sum_{j=1,\ j\neq i}^{k} m_{ij}x_j\right| \leqslant \sum_{j=1,\ j\neq i}^{k} |m_{ij}||x_j|,$$

soit en divisant par $|x_i|$ qui est non nul, $|\lambda - m_{ii}| \leqslant \sum\limits_{j=1,\ j\neq i}^{k} |m_{ij}|\dfrac{|x_j|}{|x_i|}$. Il reste à remarquer que $|x_j| \leqslant |x_i|$ pour conclure que

$$|\lambda - m_{ii}| \leqslant \sum_{j=1,\ j\neq i}^{k} |m_{ij}| \Leftrightarrow \lambda \in D_i.$$

Comme l'indice $i$ n'est pas connu, la conclusion est $\lambda \in \bigcup\limits_{i=1}^{k} D_i$.

Si 0 est une valeur propre de $A$, alors il existe un indice $i$ tel que $0 \in D_i$ où $D_i$ est le disque de centre $a_{ii}$ de rayon $\sum\limits_{j=1,\ j\neq i}^{n-1} |a_{ij}|$, c'est dire que $|0 - a_{ii}| \leqslant \sum\limits_{j=1,\ j\neq i}^{n-1} |a_{ij}|$. Puisque $A$ est à diagonale

dominante stricte, cette inégalité est impossible. Donc 0 n'est pas valeur propre de $A$ et alors $A$ est inversible. Le système $AY' = b$ admet bien une solution unique.

**5.** Nous supposons cette fois que $y'_0$ et $y'_n$ sont inconnues. Il s'agit donc d'introduire deux équations supplémentaires dans le système précédent. Graphiquement des propositions comme $y'_0 = 0$ et $y'_n = 0$ ou $y''_0 = 0$ et $y''_n = 0$ ne sont pas satisfaisantes. Par ailleurs, on perd l'erreur d'approximation en $h^4$. On pourra s'en rendre compte numériquement. En introduisant les deux conditions de continuité $\mathcal{C}^3$ aux points $x_1$ et $x_{n-1}$, on va voir que le système garde une solution unique. Par ailleurs l'étude numérique à suivre montrera que l'erreur reste en $h^4$.

Nous reprenons la méthode utilisée précédemment en calculant les dérivées 3ièmes des fonctions de la première question : $B_0^{(3)}(t) = -6$, $B_1^{(3)}(t) = 18$, $B_2^{(3)}(t) = -18$, $B_3^{(3)}(t) = 6$, si bien que $\dfrac{d^3p}{dx^3}(x) = \dfrac{6}{h^3}(-\alpha_0 + 3\alpha_1 - 3\alpha_2 + \alpha_3)$. Si nous écrivons que $S^{(3)}(x_{1+}) = S^{(3)}(x_{1-})$, comme dans la quatrième question, nous obtenons

$$\begin{aligned}
& -\alpha_0^i + 3\alpha_1^i - 3\alpha_2^i + \alpha_3^i = -\alpha_0^{i-1} + 3\alpha_1^{i-1} - 3\alpha_2^{i-1} + \alpha_3^{i-1} \\
\Leftrightarrow\quad & -y_1 + 3(y_1 + \frac{h}{3}y'_1) - 3(y_2 - \frac{h}{3}y'_2) + y_2 = -y_0 + 3(y_0 + \frac{h}{3}y'_0) - 3(y_1 - \frac{h}{3}y'_1) + y_1 \\
\Leftrightarrow\quad & y'_0 - y'_2 = \frac{2}{h}(2y_1 - y_0 - y_2).
\end{aligned}$$

En écrivant $y'_0$ en fonction des autres données et en l'introduisant dans l'équation $y'_0 + 4y'_1 + y'_2 = \dfrac{3}{h}(y_2 - y_0)$, on obtient l'équation :

$$4y'_1 + 2y'_2 = \frac{1}{h}(-4y_1 - y_0 + 5y_2).$$

De la même façon, les continuités $\mathcal{C}^2$ et $\mathcal{C}^3$ en $x_{n-1}$ donnent les 2 équations

$$y'_{n-2} - y'_n = -\frac{2}{h}(y_{n-2} - 2y_{n-1} + y_n) \quad \text{et} \quad 4y'_{n-1} + 2y'_{n-2} = \frac{1}{h}(-5y_{n-2} + 4y_{n-1} + y_n).$$

D'où le système proposé en (4.3).

**6.** Étude numérique d'erreur pour les splines. Programmer un fichier `f1.m` contenant la fonction $f$.

splin2.m

```
clear
tn=[3 5 7 10 15 20 35 50 75 100];
a=0;b=1;
nomf='f1';
for i=1:length(tn)
    n=tn(i);
    h=(b-a)/(n+1);
    x=a:h:b;
    y=feval(nomf,x);
    xabsc=a:(b-a)/1000:b;
    yord=spline(x,y,xabsc);
    taberr(i)=norm(feval(nomf,xabsc)-yord,inf);
end;
plot(log(tn+1),log(taberr));
xlabel('log(n+1)');
ylabel('log(erreur)');
a=polyfit(log(tn+1),log(taberr),1);
title(['pente de la droite de regression : ',num2str(a(1))])
```

### Ce qu'il faut retenir de cet exercice

1. Pour l'étude d'erreur, il s'agit à nouveau de faire varier $n$. On construit un tableau `tabn`, indicé par $i$ ; les erreurs successives se rangent dans un tableau et pour un $n$ donné, $n = tabn(i)$, l'erreur correspondante est en $taberr(i)$.

2. L'étude de $\log(err)$ en fonction de $\log(n+1)$ permet de trouver une puissance $\alpha$, ici environ 4, telle que $\log(err) \simeq -\alpha \log(n+1) + c_1$. On en déduit que $err \simeq \dfrac{c}{(n+1)^\alpha}$.

# 5

# Valeurs approchées d'intégrales

## RAPPEL DE COURS

**Méthodes de quadrature élémentaire :** Soit $f$ une fonction continue sur $[-1,1]$. Soient $-1 \leqslant t_1 < t_2 < \ldots t_M \leqslant 1$, $M$ points distincts de $[-1,1]$ et $\{\alpha_j\}_{j=1,\ldots,M}$, $M$ poids. La valeur approchée de $I(f) = \displaystyle\int_{-1}^{1} f(t)dt$ est définie par $J(f) = \displaystyle\sum_{j=1}^{M} \alpha_j f(t_j)$. Notons l'erreur $E(f) = |I(f) - J(f)|$. La méthode est d'ordre $r$ si pour tout $p \in \mathbb{P}_r$, on a $I(p) = J(p)$. Dans ce cas, pour $f \in \mathcal{C}^{r+1}$, en notant $M_{r+1} = \max\limits_{t\in[-1,1]} |f^{(r+1)}(t)|$, on obtient

$$E(f) \leqslant \frac{M_{r+1}}{r!} c_r \text{ où } c_r \text{ ne dépend pas de } f.$$

Voici quelques méthodes usuelles.

1. Point milieu, méthode d'ordre 1, $J(f) = 2f(0)$, $E(f) \leqslant \dfrac{1}{3} M_2$,
2. Les formules de Newton-Cotes : on interpole $f$ aux points $t_i = -1 + 2i/k$, $i = 0, \ldots, k$ par $p_k$, le polynôme de Lagrange et $J(f) = \displaystyle\int_{-1}^{1} p_k(t)dt = \sum_{j=0}^{n} \alpha_j f(t_j)$ (cf exercice 5.1), la méthode est d'ordre $k$ si $k$ est impair et d'ordre $k+1$ si $k$ est pair. Ainsi :
3. Trapèzes, méthode d'ordre 1, $J(f) = f(-1) + f(1)$, $E(f) \leqslant \dfrac{2}{3} M_2$,
4. Simpson, méthode d'ordre 3, $J(f) = \dfrac{1}{3}(f(-1) + 4f(0) + f(1))$, $E(f) \leqslant \dfrac{16}{45} M_4$,
5. Les formules de Gauss-Legendre : pour $M$ entier, le polynôme $L_M(x) = \dfrac{d^M}{dt^M}(t^2-1)^M$ admet $M$ racines distinctes $t_i$ et $t_i \in ]-1,1[$. On interpole $f$ aux points $t_i$ par $p_{M-1}$, le polynôme

de Lagrange. Si $J(f) = \int_{-1}^{1} p_{M-1}(t)dt$ alors la méthode est d'ordre $2M-1$. Exemple pour $M=2$, $J(g) = g(-1/\sqrt{3}) + g(1/\sqrt{3})$, méthode d'ordre 3.

En posant $g(t) = f\left(\frac{a+b}{2} + t\frac{b-a}{2}\right)$, $J(g)$ est une approximation de $\int_a^b f(t)dt$.

**Méthodes composées :** Soit $f$ une fonction continue sur $[a,b]$. Soit $a = x_0 < \ldots < x_n = b$ une subdivision de $[a,b]$ et $h_i = x_{i+1} - x_i$. On approche $\int_{x_i}^{x_{i+1}} f(t)dt$ par une formule de quadrature élémentaire $J(g) = \sum_{j=1}^{M} \alpha_j g(t_j)$ où $g(t) = f\left(\frac{x_i + x_{i+1}}{2} + h_i\frac{t}{2}\right)$. Ainsi $I(f) = \int_a^b f(t)dt$ est approché par

$$Iapp(f) = \sum_{i=0}^{n-1} h_i \sum_{j=1}^{M} \alpha_j f\left(\frac{x_i + x_{i+1}}{2} + h_i\frac{t_j}{2}\right).$$

Si on note $E(f) = |Iapp(f) - I(f)|$, si la méthode de quadrature élémentaire est d'ordre $r$, si $f \in C^{r+1}([a,b])$ alors

$$E(f) \leqslant C_r(b-a)h^{r+1}M_{r+1} \text{ où } h = \max h_i,\ M_{r+1} = \max_{t\in[a,b]} |f^{(r+1)}(t)|.$$

Quelques méthodes usuelles à pas constant, $h = (b-a)/n$ et $x_i = a+ih$. Soit $x_{i+1/2} = \frac{x_i + x_{i+1}}{2}$.

1. Point milieu, $Iapp(f) = h\sum_{i=0}^{n-1} f(x_{i+1/2})$, $E(f) \leqslant \frac{(b-a)^3}{24n^2}M_2 = \frac{(b-a)}{24n^2}M_2$,
2. Trapèzes, $Iapp(f) = \frac{h}{2}\left(f(x_0) + 2\sum_{i=1}^{n-1} f(x_i) + f(x_n)\right)$, $E(f) \leqslant \frac{(b-a)^3}{12n^2}M_2$,
3. Simpson, $Iapp(f) = \frac{h}{6}\left(f(x_0) + 2\sum_{i=1}^{n-1} f(x_i) + f(x_n) + 4\sum_{i=0}^{n-1} f(x_{i+1/2})\right)$, $E(f) \leqslant \frac{(b-a)^5}{180n^4}M_4$.

Les commandes `Matlab` usuelles pour le calcul d'intégrales sont `quad` et `quadl`. À noter que si nous utilisons `quad` pour intégrer la fonction « irrégulière » $f_0$ définie par $f_0(x) = |x - 1/3|$ sur $[0,1]$, le résultat proposé est `0.27777745031875` pour une valeur exacte de $5/18$ et donc une erreur relative d'environ $3.10^{-7}$. Avec `quadl`, nous obtenons respectivement `0.27777768862790` et $9.10^{-8}$. On peut évidemment améliorer ces résultats en modifiant la tolérance qui est de $10^{-6}$ par défaut. Consulter l'aide en ligne.

# ÉNONCÉS DES EXERCICES

Dans cette partie, après un premier exercice d'applications, le deuxième exercice permet de construire la méthode des trapèzes pas à pas, puis d'étudier les erreurs théorique et numérique de cette méthode. Le suivant, méthode d'extrapolation, améliore les résultats précédents, gère un tableau et à nouveau étudie une erreur. Le dernier exercice applique la méthode des trapèzes à la résolution numérique d'une équation intégrale avec étude d'erreur.

## 5.1 Premiers calculs

### 1. Interpolation et intégration

Soient $t_1, \ldots, t_M$, $M$ points distincts de $[-1, 1]$. Pour une fonction $f$ continue sur $[-1, 1]$, on désigne par $p \in \mathbb{P}_{M-1}$ le polynôme d'interpolation de Lagrange de $f$ en les $t_j$.

(a) Montrer que si nous construisons la quadrature élémentaire $J(f) = \displaystyle\int_{-1}^{1} p(t)dt$ alors

$$J(f) = \sum_{j=1}^{M} \alpha_j f(t_j);$$

préciser les $\alpha_j$ en utilisant la base de Lagrange.

(b) Montrer que la méthode est d'ordre au moins $M - 1$.

(c) Montrer que si $f \in \mathcal{C}^M([-1, 1])$, alors

$$E(f) = \left| \int_{-1}^{1} f(t)dt - J(f) \right| \leqslant \frac{\max\limits_{t\in[-1,1]} |f^{(M)}(t)|}{M!} \int_{-1}^{1} |\Pi_M(t)|dt \text{ avec } \Pi_M(t) = \prod_{j=1}^{M}(t - t_j).$$

### 2. Ordre maximum

Soient $\alpha_1$ et $\alpha_2$ 2 réels et $t_1$ et $t_2$ 2 réels de $[-1, 1]$. On considère la quadrature élémentaire $J(f) = \alpha_1 f(t_1) + \alpha_2 f(t_2)$. Déterminer ces 4 réels pour que la quadrature soit d'ordre 3.

### 3. Majoration et erreur réelle

Déterminer un entier $n$ pour que la méthode des trapèzes composée à pas constant donne une erreur inférieure à $10^{-10}$ dans l'approximation de $\displaystyle\int_0^1 t^4 dt$. (Après la programmation de l'exercice 5.2, déterminer la valeur minimum de $n$ à l'aide de `Matlab`.) Même question pour la méthode de Simpson.

## 5.2 Méthode des trapèzes

**1. Quadrature élémentaire sur** $[a,b]$

Soit $f$ une fonction continue sur $[a,b]$ à valeurs dans $\mathbb{R}$. On cherche des valeurs approchées de $I=\int_a^b f(t)dt$. Une approximation est donnée par la méthode des trapèzes :

$$i=(b-a)\frac{f(a)+f(b)}{2}.$$

(a) Montrer que si $f\in\mathcal{C}^1[a,b]$, alors $I=i-\int_a^b f'(t)(t-\frac{a+b}{2})dt$.

(b) Montrer que $J=\int_a^b f'(\frac{a+b}{2})(t-\frac{a+b}{2})dt=0$. Puis en calculant $I-J$, en déduire que pour $f\in\mathcal{C}^2[a,b]$, l'erreur vérifie :

$$e=|I-i|\leqslant\frac{1}{12}(b-a)^3M_2 \text{ où } M_2=\sup_{t\in[a,b]}|f^{(2)}(t)|. \tag{5.1}$$

**2. Formules composées**

Soit $n\in\mathbb{N}^*$. On pose $h=(b-a)/n$ et $x_k=a+kh,\ k=0,\dots,n$. Sur chaque intervalle $[x_k,x_{k+1}]$, on approche $\int_{x_k}^{x_{k+1}} f(t)dt$ par la formule précédente et on somme.

En déduire $Iapp(h)$ nouvelle approximation de $I$ puis majorer $E(h)=|I-Iapp(h)|$.

**3. Programmation**

L'objectif est d'approcher $I=\int_a^b f(t)dt$ par la formule des trapèzes

$$Iapp(h)=\frac{h}{2}\left[f(a)+f(b)+2\sum_{k=1}^{N-1}f(a+kh)\right] \text{ où } h=\frac{b-a}{N}.$$

(a) Écrire un fichier `f.m` qui contiendra la fonction $f$, par exemple $f(x)=\exp(x)$.

```
>> f([0.5 1])
ans =
    1.6487    2.7183
```

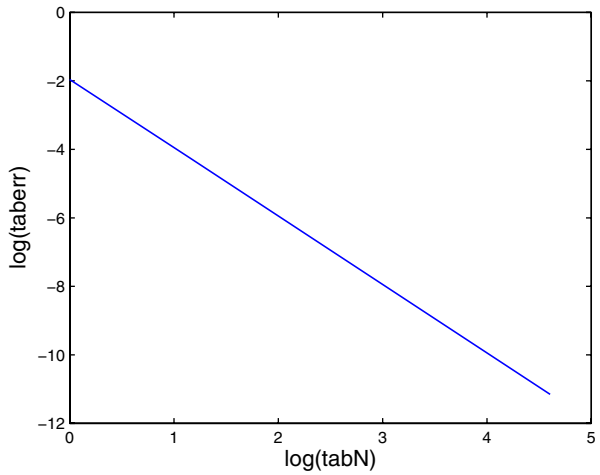
(b) Écrire un fichier `trap.m`, qui utilisera la fonction $f$ et qui permettra le calcul approché de l'intégrale par `trap(a,b,N)` et tester avec $a=0,\ b=1,\ N=10$.

```
>> trap(0,1,10)
ans =
    1.7197
```

(c) En faisant varier $N$, étudier $\log \mid Iapp(h) - I \mid$ en fonction de $\log(N)$. On tracera en particulier la courbe correspondante. Pour cette étude, on peut partir d'un tableau $tabN = [1, 2, 5, 7, 10, 15, 20, 35, 50, 75, 100]$ et construire le tableau correspondant $taberr$ puis dessiner $\log(taberr)$ en fonction de $\log(tabN)$. Sauvegarde dans `traperreur.m`.

Cette étude permet de montrer numériquement que *erreur* $\simeq C/n^{\alpha}$ ou log(*erreur*) $\simeq$ log(*c*) $- \alpha$ log(*n*). À noter que l'instruction `polyfit(log(tabN),log(taberr),1)` permet d'obtenir les coefficients de la droite de régression.

```
>> traperreur
Coefficients de la droite de regression
ans =
   -1.9976   -1.9518
```




La pente de la droite est donc d'environ $-2$ ($-1.9976$), ce qui confirme l'étude théorique.

## 5.3 Extrapolation (Méthode de Romberg)

On a vu précédemment que $E(h) = 0(h^2)$ ; on peut montrer plus précisément que dès que $f$ est suffisamment régulière, il existe $n \in \mathbb{N}$, $n > 0$ tel que

$$\int_a^b f(t)dt - Iapp(h) = \alpha_{2,2}h^2 + \alpha_{2,4}h^4 + \ldots + \alpha_{2,2n}h^{2n} + o(h^{2n}).$$

On pose $T_0(h) = Iapp(h)$, $T_{k+1}(h) = [4^{k+1}T_k(h/2) - T_k(h)]/[4^{k+1} - 1]$, pour $k = 0, 1, \ldots$
Montrer que $I - T_1(h) = O(h^4)$, $I - T_2(h) = O(h^6), \ldots, I - T_n(h) = 0(h^{2n})$.

**Programmation**

On utilise les fichiers `trap.m, f.m` précédents, on part de $h = b-a$. On fixe $n > 0$ et on remplira le tableau triangulaire $T$, puis le tableau d'erreurs. Attention avec `Matlab`, les indices de tableau commencent à 1.

| | | | | | |
|---|---|---|---|---|---|
| $T(1,1) = trap(a,b,1)$ | $T(1,2)$ | $T(1,3)$ | $\dots$ | $T(1,n)$ | $T(1,n+1) = trap(a,b,2^n)$ |
| $T(2,1) = \dfrac{4T(1,2)-T(1,1)}{4-1}$ | $T(2,2)$ | | $\dots$ | $T(2,n)$ | 0 |
| $\vdots$ | | $\dots$ | | 0 | $\vdots$ |
| $T(i+1,1) = \dfrac{4^i T(i,2)-T(i,1)}{4^i-1}$ | $\dots$ | $T(i+1,n+1-i)$ | 0 | | $\vdots$ |
| $\vdots$ | | 0 | | | $\vdots$ |
| $T(n+1,1) = \dfrac{4^n T(n,2)-T(n,1)}{4^n-1}$ | 0 | $\dots$ | $\dots$ | $\dots$ | 0 |

**1.** Construire la première ligne du tableau $T(1,:)$ correspondant aux valeurs successives de $Iapp(h), \dots, Iapp(h/2^n)$. Sauvegarde sous `romb1.m`. Test pour $a = 0, b = 1, n = 5$.

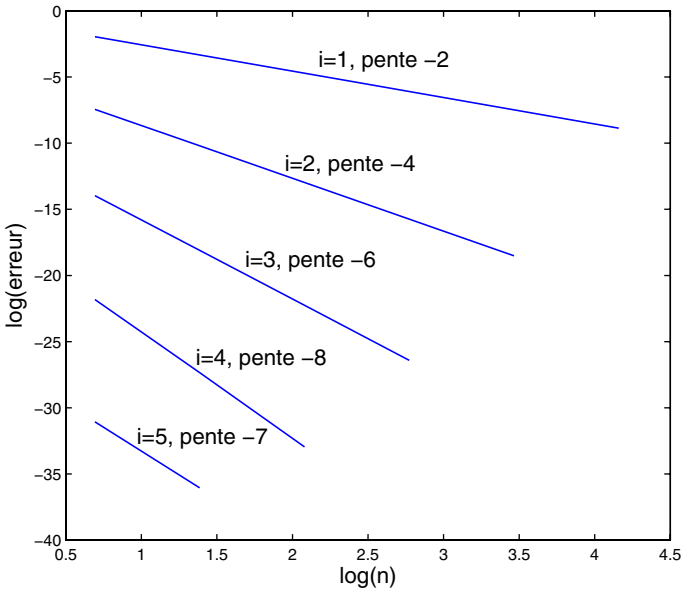
```
>> romb1
T =
    1.8591    1.7539    1.7272    1.7205    1.7188    1.7184
```

**2.** Compléter le tableau $T$. Sauvegarde dans `romb2.m`. Test pour $a = 0, b = 1, n = 5$.

```
>> romb2
T =
    1.8591    1.7539    1.7272    1.7205    1.7188    1.7184
    1.7189    1.7183    1.7183    1.7183    1.7183         0
    1.7183    1.7183    1.7183    1.7183         0         0
    1.7183    1.7183    1.7183         0         0         0
    1.7183    1.7183         0         0         0         0
    1.7183         0         0         0         0         0
```

**3.** Compléter le programme en ajoutant un tableau d'erreurs `taberr` puis étudier l'erreur en fonction de la ligne $i$. En particulier, on tracera les courbes $\log(taberr(i,:))$ en fonction de $j \times \log(2)$. En effet, pour $N = 2^{j-1}$, l'erreur attendue $taberr(i,j)$ est en $\left(\dfrac{b-a}{2^{j-1}}\right)^{2i}$. Sauvegarde dans `romb3.m`. Test pour $a = 0, b = 1, n = 5$

```
>> romb3
>> taberr(2,1:5)
ans =
  1.0e-003 *
    0.5793    0.0370    0.0023    0.0001    0.0000
```




On remarque qu'à partir de $i = 5$, on ne retrouve plus la précision attendue ; nous atteignons la limite de précision de la machine. Ainsi $\mathit{taberr}(5,2) = 2.2204 \times 10^{-16}$.

## 5.4 Équation intégrale

On souhaite déterminer une approximation $V$ de la solution $u$ de l'équation intégrale :

$$u(x) = \int_a^b K(x,t)u(t)dt + f(x), \quad x \in [a,b]$$

où $a$ et $b$ sont donnés ainsi que les fonctions $K$ et $f$ ; $K$ est appelée noyau. Pour information, l'équation de Love en électrostatique est

$$u(x) = \frac{1}{\pi}\int_{-1}^{1} \frac{1}{1+(x-t)^2}\, u(t)dt + 1\ .$$

Le calcul approché de l'intégrale se fait par une méthode des trapèzes. À partir d'un entier $N$, on construit une subdivision régulière de $[a,b]$ en définissant $h = (b-a)/N$ et $x_i = a + (i-1)h$ pour $i = 1$ à $N+1$. Ainsi $\displaystyle\int_a^b \varphi(t)dt \simeq \frac{h}{2}\big(\varphi(x_1) + 2\sum_{j=2}^{N}\varphi(x_j) + \varphi(x_{N+1})\big)$, si bien que pour chaque $i$, l'équation du problème approché s'écrit :

$$V_i = \frac{h}{2}\left(K(x_i,x_1)V_1 + 2\sum_{j=2}^{N} K(x_i,x_j)V_j + K(x_i,x_{N+1})V_{N+1}\right) + f(x_i)$$

où $V_j$ est l'approximation de $u(x_j)$, ce qui conduit au problème approché :

$$\left(I - \frac{h}{2}A_N\right) V = F$$

où $I$ est la matrice identité de $\mathbb{R}^{(N+1)\times(N+1)}$,

$F = (f(x_1), f(x_2), ..., f(x_{N+1}))^T$,

$$A_N = \begin{pmatrix} K(x_1,x_1) & 2K(x_1,x_2) & \dots & 2K(x_1,x_N) & K(x_1,x_{N+1}) \\ K(x_2,x_1) & 2K(x_2,x_2) & & \vdots & \vdots \\ \vdots & \vdots & & \vdots & \vdots \\ K(x_{N+1},x_1) & 2K(x_{N+1},x_2) & \dots & 2K(x_{N+1},x_N) & K(x_{N+1},x_{N+1}) \end{pmatrix},$$

et $V = (V_1, V_2, ..., V_{N+1})^T \in \mathbb{R}^{N+1}$ est l'inconnue.

Dans la suite du problème, on suppose que la fonction $K$ vérifie $K(x,t) = k(x-t)$ où $k$ est une fonction définie et continue sur $\mathbb{R}$.

**1.** Écrire un premier programme (sans boucle) qui étant donnés $a, b$ et $N$ construit le vecteur $x$ puis la matrice $B_N \in \mathbb{R}^{(N+1)\times(N+1)}$ définie par

$$B_N = (x_i - x_j) = \begin{pmatrix} 0 & x_1 - x_2 & \cdots & x_1 - x_N & x_1 - x_{N+1} \\ x_2 - x_1 & 0 & & \vdots & \vdots \\ \vdots & \vdots & & 0 & x_N - x_{N+1} \\ x_{N+1} - x_1 & x_{N+1} - x_2 & & x_{N+1} - x_N & 0 \end{pmatrix}$$

Sauvegarde sous `ei1.m`. Test $a = -1$, $b = 1$, $N = 3$, afficher $BN$.

```
>> ei1
BN =
         0   -0.6667   -1.3333   -2.0000
    0.6667         0   -0.6667   -1.3333
    1.3333    0.6667         0   -0.6667
    2.0000    1.3333    0.6667         0
```

**2.** Écrire deux fichiers de type function qui étant donné $t$, réel, vecteur ou tableau puis calcule $k(t)$ et $f(t)$. Sauvegarde sous `k.m` et `f1.m`. Tester avec `f([1 2]`, idem pour `k`.

Exemple : $k(t) = t^2$, $f(t) = \sin(\pi t) + \dfrac{4t}{\pi}$.

**3.** Compléter `ei1.m` pour construire les matrices $C_N = (k(x_i - x_j))$ puis $A_N$. Sauvegarde sous `ei2.m`.

Test : $a = -1$, $b = 1$, $N = 3$ ; afficher $A_N$.

```
>> ei2
AN =
         0    0.8889    3.5556    4.0000
    0.4444         0    0.8889    1.7778
    1.7778    0.8889         0    0.4444
    4.0000    3.5556    0.8889         0
```

**4.** Compléter `ei2.m` en construisant le second membre puis la solution $V$ de

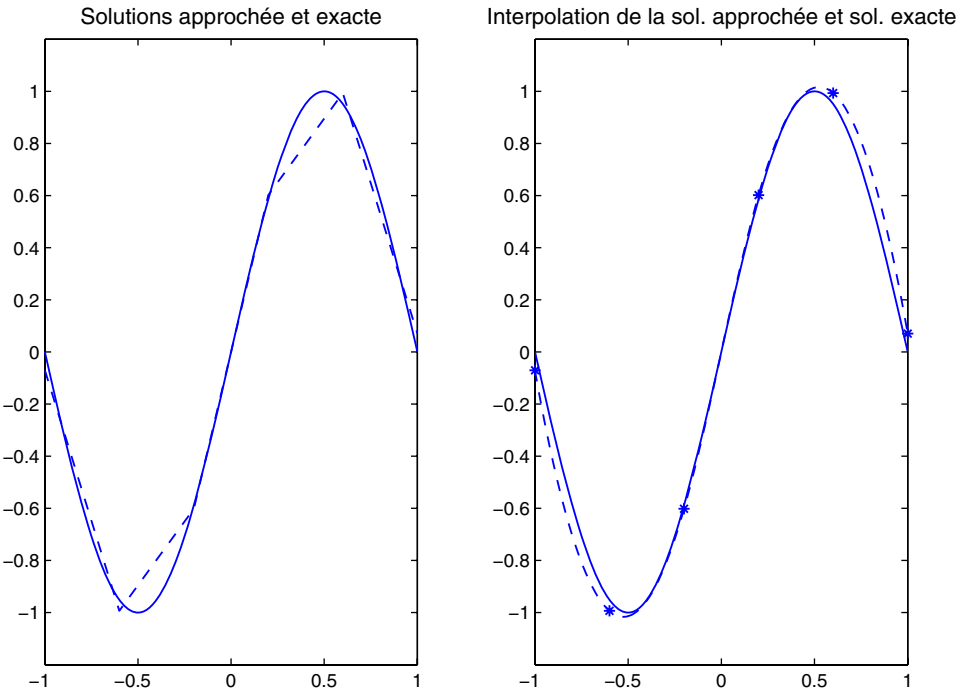
$$(I - \frac{h}{2}A_N)V = F.$$

Sauvegarde sous `ei3.m`. Test : $a = -1,\ b = 1,\ N = 5$ ; afficher $V$.

```
>> ei3
V =
   -0.0705
   -0.9934
   -0.6019
    0.6019
    0.9934
    0.0705
```

**5.** Calculer l'erreur : $err = \|U - V\|_\infty = \max_{i=1,\ldots,\ N+1}(|U_i - V_i|)$ avec $U_i = u(x_i)$ puis tracer la courbe continue affine par morceaux passant par les points $(x_i, V_i)$, $i = 1$ à $N + 1$ ainsi que la solution exacte donnée par $u(t) = \sin(\pi t)$. On pourra remplacer la courbe affine par morceaux par une spline cubique (cf aide en ligne). Sauvegarde sous `ei4.m`. Test : $a = -1,\ b = 1,\ N = 5$ ; affichage de $err$ et graphique.

```
>> ei4
err =
    0.0705
```




**6.** Étudier l'erreur $\|U - V\|_\infty$ quand $N$ varie. On partira du tableau de valeurs $tN = [5, 7, 10, 15, 20, 30, 50, 75, 100, 150, 200]$ pour construire le tableau $taberr$ correspondant, puis tracer la courbe $\log(taberr)$ en fonction de $\log(tN)$ et utiliser `polyfit`. Sauvegarde sous `ei5.m`. Test : $a = -1,\ b = 1$.

```
>> ei5
Coefficients de la droite de regression
ans =
   -1.9966    0.5699
```

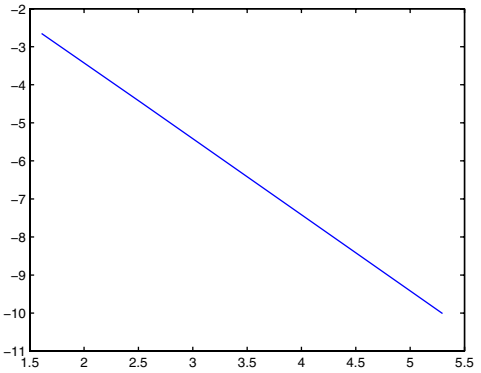
On trouve une erreur en $N^{-2}$ (pente de la droite de régression $-1.9966$).

**7.** Reconstruire un nouveau schéma et faire une étude d'erreur pour un calcul approché de l'intégrale par la méthode de Simpson :

$$\int_a^b \varphi(t)dt \simeq \frac{h}{3}(\varphi(x_1) + \varphi(x_{N+1}) + 2\sum_{k=1}^{N'-1} \varphi(x_{2k+1}) + 4\sum_{k=1}^{N'} \varphi(x_{2k}))$$

où $N = 2N'$ et $h = (b-a)/N$. Sauvegarde sous `ei6.m` (éventuellement `ei7.m`). Exemple : $Np = [3, 4, 5, 6, 10, 15, 37, 50, 75, 100]$, $tN = 2 \times tNp$, $a = -1,\ b = 1$.

```
>> ei7
ans =
   -4.0277    1.6790
```




L'erreur est en $h^4$ avec une pente de la droite de régression de $-4.0277$.

## DU MAL À DÉMARRER ?

### 5.1 Interpolation et intégration

**1.** Utiliser les résultats de l'exercice 4.2.

**2.** Écrire quatre équations vérifiées par les réels.

### 5.2 Méthode des trapèzes

Utiliser une intégration par parties.

Le théorème des accroissements finis peut être utile.

Pour la formule composée, majorer la valeur absolue par la somme des valeurs absolues.

### 5.3 Extrapolation

En fait, ce n'est pas si compliqué... Écrire les formules pour $k = 0$, c'est-à-dire $T_1(h)$ puis $T_1(h) - \int_a^b f(t)dt$.

# CORRIGÉS DES EXERCICES

## 5.1 Premiers calculs

### 1. Interpolation et intégration

Si $p$ est l'interpolant de Lagrange de $f$ en les $t_j$, alors $p(t) = \sum_{j=1}^{M} \ell_j(t) f(t_j)$ où $\ell_j(t) = \prod_{\substack{k=1 \\ k\neq j}}^{M} \frac{x - x_k}{x_j - x_k}$. Donc par linéarité, $\int_{-1}^{1} p(t)dt = \sum_{j=1}^{M} f(t_j) \int_{-1}^{1} \ell_j(t)dt = \sum_{j=1}^{M} \alpha_j f(t_j)$.

Si $q \in \mathbb{P}_{M-1}$ alors, par unicité, son interpolant de Lagrange aux point $t_j$ coïncide avec lui même. Donc $\int_{-1}^{1} q(t)dt = J(q)$. La méthode est d'ordre au moins $M-1$.

À l'exercice 4.2, nous avons vu que si $f \in \mathcal{C}^M([-1,1])$, et si $p$ désigne son interpolant de Lagrange aux points $t_j$ alors pour tout $t$ de $[-1,1]$, il existe $\xi \in ]-1,1[$ tel que $f(t) - p(t) = \frac{f^{(M)}(\xi)}{M!}\Pi_M(t)$ avec $\Pi_M(t) = \prod_{j=1}^{M}(t - t_j)$. En majorant $|f^{(M)}(\xi)|$ par $\max_{t\in[-1,1]} |f^{(M)}(t)|$ puis en intégrant entre $-1$ et 1, il vient $E(f) = \left| \int_{-1}^{1} f(t)dt - J(f) \right| \leqslant \frac{\max_{t\in[-1,1]} |f^{(M)}(t)|}{M!} \int_{-1}^{1} |\Pi_M(t)|dt$.

### 2. Ordre maximum

Soit $J(f) = \alpha_1 f(t_1) + \alpha_2 f(t_2)$ une quadrature élémentaire. Pour qu'elle soit d'ordre 3, elle doit être exacte pour tous les polynômes de degré inférieur ou égal à 3. Par linéarité de l'intégrale et de la quadrature, il faut et suffit qu'elle soit exacte pour une base de $\mathbb{P}_3$. En prenant par exemple la base canonique, il vient :

$$
\begin{aligned}
&(1) \quad \alpha_1 + \alpha_2 = 2,\\
&(2) \quad \alpha_1 t_1 + \alpha_2 t_2 = 0,\\
&(3) \quad \alpha_1 t_1^2 + \alpha_2 t_2^2 = 2/3,\\
&(4) \quad \alpha_1 t_1^3 + \alpha_2 t_2^3 = 0,
\end{aligned}
$$

Si $\alpha_1 = 0$ alors $\alpha_2 = 2$ par (1) et $t_2 = 0$ par (2) ce qui n'est pas compatible (3). Si $t_1 = 0$ alors (3) et (4) conduisent à une impossibilité. Donc $\alpha_1 \neq 0$ et $t_1 \neq 0$. De même $\alpha_2 \neq 0$ et $t_2 \neq 0$.

Si $t_1 = \pm 1$, par différence de (2) et (4), on obtient $\alpha_2 t_2 (1 - t_2)^2 = 0$ donc $t_2 = \pm 1$. Mais alors (1) et (3) sont incompatibles. Donc $t_1, t_2 \notin \{-1, 1\}$.

On effectue la différence (2) moins (4) pour obtenir $\alpha_1 t_1 (1 - t_1^2) + \alpha_2 t_2 (1 - t_2^2) = 0$ ou encore $\frac{\alpha_2 t_2 (1 - t_2^2)}{\alpha_1 t_1 (1 - t_1^2)} = -1$. Sachant que par (2), $\frac{\alpha_2 t_2}{\alpha_1 t_1} = -1$, il vient $\frac{(1 - t_2^2)}{(1 - t_1^2)} = 1$ soit $t_1^2 = t_2^2$.

Si $t_1 = t_2$, alors par (1) et (2), $t_1 = 0$ ce qui est impossible. Reste le cas $t_1 = -t_2$. Par (1) et (2), il vient $\alpha_1 = \alpha_2 = 1$. (4) est ainsi vérifiée. (3) donne $t_1 = -t_2 = \pm 1/\sqrt{3}$. On retrouve la méthode de Gauss-Legendre, seule quadrature élémentaire à 2 points d'ordre 3.

### 3. Majoration et erreur réelle

Pour la méthode des trapèzes composée à pas constant, l'erreur $E(f)$ est majorée par $\dfrac{(b-a)^3}{12n^2}M_2$. Si $f(x) = x^4$ sur $[0,1]$, $M_2 = 12$ et $(b-a)^3 = 1$. Donc il suffit que $n \geqslant 10^5$, pour avoir une précision de $10^{-10}$ dans l'approximation de $\displaystyle\int_0^1 t^4 dt$. Il s'agit d'une condition suffisante. À l'aide de `Matlab`, on peut montrer que $n \geqslant 57736$ suffit. Il reste tout de même 57737 évaluations de la fonction $f$.

```
>> format long
>> trap(0,1,57736)-1/5
ans =
    9.999667760496322e-011
>> trap(0,1,57735)-1/5
ans =
    1.000005633855494e-010
```

On reprend le raisonnement pour la méthode de Simpson. $M_4 = 24$ et $(b-a) = 1$ ; or $E(f) \leqslant \dfrac{M^4(b-a)^5}{180n^4}$ donc $E(f) \leqslant 10^{-10}$ dès que $\left(\dfrac{2.10^{10}}{15}\right)^{1/4}$, soit $n \geqslant 192$ ou encore 385 évaluations de la fonction $f$.

## 5.2 Méthode des trapèzes

Pour évaluer l'erreur $e$, on part de $\displaystyle\int_a^b f(t)dt$ qu'on intègre par parties et on utilise $\displaystyle\int_a^b f'\left(\frac{a+b}{2}\right)\left(t - \frac{a+b}{2}\right)dt = 0.$

$$\begin{aligned}
\int_a^b f(t)dt &= \left[f(t)\left(t - \frac{a+b}{2}\right)\right]_a^b - \int_a^b f'(t)\left(t - \frac{a+b}{2}\right)dt \\
&= \frac{f(a)+f(b)}{2}(b-a) - \int_a^b f'(t)\left(t - \frac{a+b}{2}\right)dt \\
&+ \int_a^b f'\left(\frac{a+b}{2}\right)\left(t - \frac{a+b}{2}\right)dt \\
&= \frac{f(a)+f(b)}{2}(b-a) + \int_a^b \left(f'\left(\frac{a+b}{2}\right) - f'(t)\right)\left(t - \frac{a+b}{2}\right)dt
\end{aligned}$$

D'où $|I-i| \leqslant \int_a^b \left| f'\left(\frac{a+b}{2}\right) - f'(t)\right| . \left| t - \frac{a+b}{2}\right| dt$. Puisque $f \in \mathcal{C}^2[a,b]$, $\left| f'\left(\frac{a+b}{2}\right) - f'(t)\right| \leqslant M_2 \left| \frac{a+b}{2} - t\right|$ par le théorème des accroissements finis. Alors $|I-i| \leqslant \int_a^b M_2 \left(t - \frac{a+b}{2}\right)^2 dt$ et en intégrant, $e \leqslant \frac{M_2}{12}(b-a)^3$. On peut proposer une autre démonstration en utilisant l'interpolation de Lagrange et le résultat d'erreur (4.1).

La formule composée s'obtient en additionnant les approximations successives de $\int_{x_k}^{x_{k+1}} f(t)dt$, soit $\frac{f(x_k)+f(x_{k+1})}{2}h$ pour $k = 0, \ldots, n-1$. D'où la nouvelle formule d'approximation

$$I(h) = \frac{h}{2}\left( f(x_0) + f(x_n) + 2\sum_{k=1}^{n-1} f(x_k)\right).$$

L'erreur globale, quant à elle, se majore par la somme des erreurs sur chaque intervalle $[x_k, x_{k+1}]$. Sachant que $\sup_{[x_k,x_{k+1}]} |f''(t)| \leqslant \sup_{[a,b]} |f''(t)| = M_2$, il vient

$$E(h) = |I - I(h)| \leqslant n\frac{M_2}{12}h^3 = \frac{M_2(b-a)^3}{12}.\frac{1}{n^2}.$$

On peut montrer que la majoration en $1/n^2$ est optimale à l'aide d'un polynôme du second degré. On le verra aussi numériquement à l'aide de l'étude d'erreur ci-dessous.

`f.m`

```
function y=f(x);
y=exp(x);
```

`trap.m`

```
function Ih=trap(a,b,N)
h=(b-a)/N;
x=a:h:b;
y=f(x);
Ih=h*(y(1)+y(N+1)+2*sum(y(2:N)))/2;
```

`traperreur.m`

```
a=0;b=1;
tabN=[1,2,5,7,10,15,20,35,50,75,100];
Iex=exp(1)-1;
for i=1:length(tabN)
    taberr(i)=abs(trap(a,b,tabN(i))-Iex);
end;
```

```
plot(log(tabN),log(taberr));
xlabel('log(tabN)');ylabel('log(taberr)');
disp('Coefficients de la droite de regression')
polyfit(log(tabN),log(taberr),1)
```

## 5.3 Méthode de Romberg

Par construction $T_1(h) = (4T_0(h/2) - T_0(h)]/[4-1]$. Nous avons alors

$$\begin{aligned} I - T_1(h) &= \left[4(I - T_0(h/2)) - (I - T_0(h)\right]/(4-1) \\ &= \left[4(\alpha_{22}(h/2)^2 + \alpha_{24}(h/2)^4 + ... + \alpha_{22n}(h/2)^{2n} + o(h^{2n})) \right. \\ &\quad \left. -(\alpha_{22}h^2 + \alpha_{24}h^4 + ... + \alpha_{22n}h^{2n} + o(h^{2n})\right]/3 \\ &= \alpha_{44}h^4 + ... + \alpha_{42n}h^{2n} + o(h^{2n}). \end{aligned}$$

L'erreur est en $h^4$. De la même façon, on montre que l'erreur pour $I - T_2(h)$ est en $h^6$ etc... $I - T_n(h) = O(h^{2n+2})$.

romb123.m

```
close all;
a=0;b=1;
n=5;
Iex=exp(1)-1;
%romb1
for j=0:n
    T(1,j+1)=trap(a,b,2^j);
end;

%romb2
for i=1:n
    mult=4^i;divis=mult-1;
    for j=1:n+1-i
        T(i+1,j)=(mult*T(i,j+1)-T(i,j))/divis;
    end;
end;

%romb3
taberr=abs(T-Iex);
taberr(2,1:5)
for i=1:n
    plot((1:n+2-i)*log(2),log(taberr(i,1:n+2-i)));
    i
    polyfit((1:n+2-i)*log(2),log(taberr(i,1:n+2-i)),1)
    hold on
end;
```

### Ce qu'il faut retenir de cet exercice

Cette fois, on utilise plusieurs boucles. Pour la première ligne, difficile de programmer la méthode des trapèzes pour un tableau de valeurs de $n$. Ensuite la ligne $i+1$ peut être construite

uniquement en connaissant le ligne $i$. Par contre, on pourrait améliorer la programmation pour obtenir toutes les colonnes simultanément.

## 5.4 Équation intégrale

ei4.m

```
clear
a=-1;b=1;
N=5;
h=(b-a)/N;
x=a:h:b;
BN=x'*ones(1,N+1)-ones(N+1,1)*x;
CN=k(BN);
AN=CN;
AN(:,2:N)=2*AN(:,2:N);
F=f1(x)';
V=(eye(N+1)-h*AN/2)\F;
err=norm(V-u(x)',inf)
t=a:0.005:b;
subplot(1,2,1)
plot(x,V,'--b',t,u(t),'b')
axis([-1 1 -1.2 1.2])
title('Solutions approchée et exacte')
subplot(1,2,2)
plot(x,V,'*b',t,spline(x,V,t),'--b',t,u(t),'b');
axis([-1 1 -1.2 1.2])
title('Interpolation de la sol. approchée et sol. exacte')
```

### Ce qu'il faut retenir de cet exercice

Comme dans les exercices sur la dérivation approchée du chapitre précédent, les matrices se construisent sans boucle. Pour obtenir des tableaux partiels, on peut revoir le chapitre 1.

ei5.m

```
clear
a=-1;b=1;
tN=[5 7 10 15 20 30 50 75 100 150 200];
for i=1:length(tN)
    N=tN(i);
    h=(b-a)/N;
    x=a:h:b;
    BN=x'*ones(1,N+1)-ones(N+1,1)*x;
    CN=k(BN);
    AN=CN;
    AN(:,2:N)=2*AN(:,2:N);
    F=f1(x)';
    V=(eye(N+1)-h*AN/2)\F;
    taberr(i)=norm(V-u(x)',inf);
end;
```

```
plot(log(tN),log(taberr));
disp('Coefficients de la droite de regression')
polyfit(log(tN),log(taberr),1)
```

### Ce qu'il faut retenir de cet exercice

On trouve une erreur de l'ordre de $1/N^2$ avec une approximation par la méthode des trapèzes qui est aussi en $1/N^2$, mais entre les deux résultats, la résolution du système $(I - hA_N/2)V = F$ aurait pu amplifier les perturbations (voir chapitre 3). Il est alors intéressant de calculer le conditionnement de la matrice $(I - hA_N/2)$ lorsque $N$ varie ; à partir de $N = 50$ il semble garder une valeur de l'ordre de 110.

ei7.m

```
clear
a=-1;b=1;
tNp=[ 3 4 5 6 10 15 37 50 75 100];
tN=2*tNp;
for i=1:length(tN)
    N=tN(i);
    h=(b-a)/N;
    x=a:h:b;
    BN=x'*ones(1,N+1)-ones(N+1,1)*x;
    CN=k(BN);
    AN=CN;
    AN(:,2:2:N)=4*AN(:,2:2:N);
    AN(:,3:2:N-1)=2*AN(:,3:2:N-1);
    F=f1(x)';
    V=(eye(N+1)-h*AN/3)\F;
    taberr(i)=norm(V-u(x)',inf);
end;
plot(log(tN),log(taberr));
polyfit(log(tN),log(taberr),1)
```

### Ce qu'il faut retenir de cet exercice

Mêmes remarques que précédemment sur la gestion des matrices sans boucle et sur le conditionnement.

# 6 Moindres carrés

## RAPPEL DE COURS

Soit $A \in \mathbb{R}^{n\times m}$ et $b \in \mathbb{R}^n$ avec $n \geqslant m$. Le système $AX = b$ n'a pas toujours de solution (plus d'équations que d'inconnues). On peut alors chercher à minimiser $J(\alpha) = \|A\alpha - b\|_2$ où $\|U\|_2 = \left(\sum_{i=1}^{n} u_i^2\right)^{1/2}$.

*Si $A \in \mathbb{R}^{n\times m}$ avec $n \geqslant m$ alors $A = QR$ où $R$ est une matrice triangulaire supérieure de $\mathbb{R}^{m\times m}$ et $Q$ une matrice de $\mathbb{R}^{n\times m}$ telle que $Q^TQ = I$. Si $A$ est de rang $m$ alors $J$ admet un minimum unique atteint en $\bar{\alpha}$ solution du système linéaire $Rx = Q^Tb$ ou du système $A^TA\bar{\alpha} = A^Tb$.*

**Application :** Soient $x_1, x_2, \ldots, x_n$, $n$ réels distincts, $b$ un vecteur de $\mathbb{R}^n$ et $\{\varphi_1, \varphi_2, \ldots, \varphi_m\}$, $m$ fonctions linéairement indépendantes définies sur $\mathbb{R}$ (ou une partie de $\mathbb{R}$) avec $m \leqslant n$. Pour déterminer une fonction $\varphi = \sum_{j=1}^{m} \alpha_j\varphi_j$ minimisant $J(\alpha) = \left(\sum_{i=1}^{n}(\varphi(x_i) - y_i)^2\right)^{1/2}$, on construit $A \in \mathbb{R}^{n\times m}$ telle que $a_{ij} = \phi_j(x_i)$ et on minimise $\|A\alpha - b\|_2$. Si $A$ est de rang $m$, la solution unique est obtenue en $\bar{\alpha}$ solution de $A^TA\bar{\alpha} = A^Tb$.

Le cas le plus connu est l'approximation par une droite de régression où $\varphi_1(x) = 1$ et $\varphi_2(x) = x$. On cherche $\varphi(x) = \alpha_1 + \alpha_2 x$ minimisant les carrés des écarts.

Nous commençons par redémontrer comment obtenir une approximation au sens des moindres carrés. Un exemple de droite et de parabole de régression est ensuite proposé. L'instruction `polyfit` de `Matlab` permet à partir de données $(x_i, y_i)_{i=1,\ldots,n}$, d'obtenir les coefficients polynômiaux de l'approximation $y(x)$ au sens des moindres carrés.

Mais les fonctions de base les plus intéressantes ne sont pas forcément les polynômes. On verra l'intérêt de fonctions périodiques dans l'exercice 6.3. Pour le filtre de Savitsky-Golay (instruction `sgolayfilt`), nous décrivons la méthode ; elle permet de montrer l'importance du choix d'une bonne base de polynômes et de faire quelques calculs sur les matrices ; nous proposons aussi une programmation qui rajoute quelques paramètres par rapport à l'instruction standard.

Enfin nous terminons avec la recherche d'une solution approchée d'une équation différentielle par une méthode de moindres carrés.

# ÉNONCÉS DES EXERCICES

## 6.1 Mise en équation

On rappelle que pour $u \in \mathbb{R}^n$, $\|u\|_2 = (u^T \times u)^{1/2}$ où $u^T \times v$ calcule le produit scalaire $u.v = \sum_{i=1}^{n} u_i v_i$, avec $u, v \in \mathbb{R}^n$. Le choix de cette norme n'est pas anodin et on verra que le produit scalaire est indispensable pour utiliser les projections orthogonales.

Soient $x_1 < x_2 < \ldots < x_n$, $n$ abscisses distinctes, $b$ un vecteur de $\mathbb{R}^n$ et $\{\varphi_1, \varphi_2, \ldots, \varphi_m\}$, $m$ fonctions linéairement indépendantes définies sur $\mathbb{R}$ avec $m \leqslant n$. On cherche une fonction $\varphi = \sum_{j=1}^{m} \alpha_j \varphi_j$ approchant au mieux les données i.e $b_i \simeq \varphi(x_i)$. Précisons : il s'agit de trouver un ou des vecteurs $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_m)^T \in \mathbb{R}^M$ minimisant $J(\alpha) = \big(\sum_{i=1}^{n} (\varphi(x_i) - b_i)^2\big)^{1/2}$.

Soit $A$ la matrice rectangulaire de $\mathbb{R}^{n \times m}$ dont les coefficients sont définis par $a_{ij} = \varphi_j(x_i)$ pour $i = 1, \ldots, n,\ j = 1, \ldots, m$.

**1.** Montrer que $\big(\varphi(x_1), \ldots, \varphi(x_n)\big)^T = A(\alpha_1, \ldots, \alpha_m)^T$.

**2.** En déduire que $J(\alpha) = \|A\alpha - b\|_2$.

**3.** Montrer que si $E$ est un sous-espace vectoriel de $\mathbb{R}^n$, le minimum de $\|u - b\|_{u \in E}$ est atteint en $\bar{u}$ projection orthogonale de $b$ sur $E$ définie par $\bar{u} \in E$ et $\forall v \in E$, $v^T \times (\bar{u} - b) = 0$. En fait il suffit d'avoir l'égalité pour une base de $E$. On en déduit que le minimum de $J(\alpha)$ est atteint lorsque le vecteur $(\varphi(x_1), \ldots, \varphi(x_n))^T$ est le projeté orthogonal de $b$ sur $\mathrm{Im}(A)$.

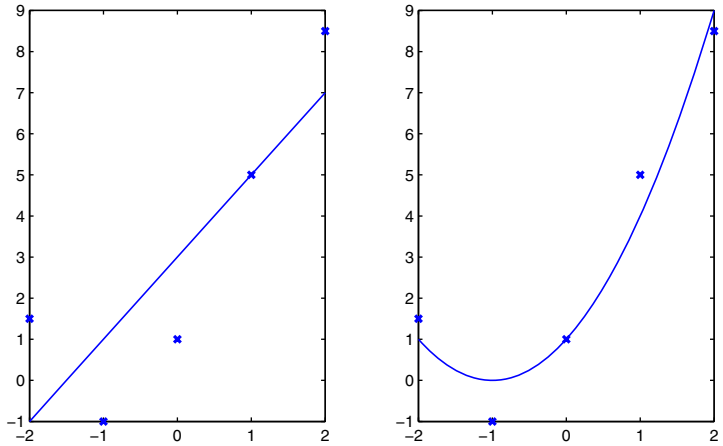
**4.** Montrer que le minimum de $J(\alpha)$ est atteint en $\bar{\alpha}$ vérifiant : $A^T A\bar{\alpha} = A^T b$ qui admet une solution unique dès que $\mathrm{rang}(A) = m$ .

## 6.2 Droite et parabole de régression

On part des données suivantes :

| $i$ | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| $x$ | $-2$ | $-1$ | 0 | 1 | 2 |
| $y$ | $3/2$ | $-1$ | 1 | 5 | $17/2$ |

Déterminer la droite puis la parabole de régression i.e minimiser $\sum_{i=1}^{5} (y_i - \alpha_1 - \alpha_2 x_i)^2$ puis $\sum_{i=1}^{5} (y_i - \alpha_1 - \alpha_2 x_i - \alpha_3 x_i^2)^2$. Vérification sur `Matlab`.



## 6.3 Signal périodique

On considère un signal $y$ périodique échantillonné sur $[-1, 1]$. Pour l'instant $y$ est supposé impair et donné aux abscisses croissantes $x_k$, $k = 1, \ldots, N$ avec $x_1 = -1$ et $x_N = 1$. En fait, le signal a été bruité et on a reçu le signal $yb$ qu'on souhaite lisser par une méthode de moindres carrés.

Pour lisser le signal, étant donné un entier $M$, on va minimiser

$$E(P) = \sum_{k=1}^{N} (yb_k - P(x_k))^2 \text{ où } P(t) = \sum_{p=1}^{M} u_p \sin(p\pi t).$$

On cherche donc $U = (u_1, \ldots, u_M)^T$ solution du système $A^T AU = A^T yb$ où $A$ est la matrice rectangulaire $N \times M$ définie par $a_{ij} = \sin(j\pi x_i)$ et $yb$ une colonne à $N$ composantes.

On obtient le signal bruité à l'aide du programme ci-dessous, `signal1.m` qui étant donné $N$ renvoie 3 vecteurs $x$, $y$, $yb$, avec $x$, les abscisses, $y$ le signal exact qui servira en fin de programme à comparer au résultat du lissage, et $yb$ le signal bruité.

```
function [x,y,yb]=signal1(N);
x=sort(2*rand(N,1)-1);
x(1)=-1;x(N)=1;
y=sin(6*pi*x);
yb=y+(2*rand(N,1)-1)*0.6;
```

**1.** Construire un premier programme sauvegardé sous `SP1.m` qui prend $N$ et $M$ en paramètres d'entrée, appelle `signal1(N)`, construit et affiche la matrice $A$. On commencera par construire (sans boucle) la matrice $B$ telle que $b_{ij} = x_i \times j$. Test `SP1(6,3)`.

Attention du fait du tirage aléatoire, les résultats seront différents.

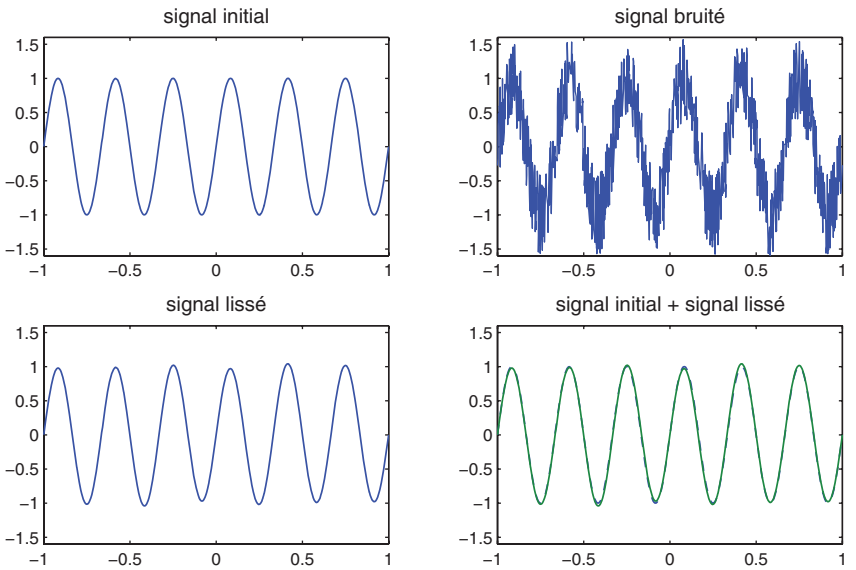
```
>> SP1(6,3)
A =
    -0.0000     0.0000    -0.0000
    -0.8714     0.8550     0.0326
     0.4297     0.7760     0.9718
     0.6085     0.9658     0.9242
     0.6830     0.9977     0.7747
     0.0000    -0.0000     0.0000
```

**2.** Compléter le 1er programme en `SP2.m` en résolvant le système $A^T AU = A^T yb$ ; ne plus afficher $A$ mais afficher $U$. Test `SP2(1000,8)`.

```
>> SP2(1000,8)
U =
     0.0152
    -0.0139
     0.0056
     0.0262
     0.0072
     1.0043
     0.0138
    -0.0161
```

Naturellement comme les points $x_k$ sont aléatoires ainsi que le bruit, les valeurs peuvent être légèrement différentes, mais la sixième est proche de 1, les autres de 0.

**3.** Compléter le 2ème programme en `SP3.m` par le calcul du signal lissé $y\ell_k = \sum_{p=1}^{M} u_p \sin(p\pi x_k)$, $k = 1$ à $N$ (sans boucle), puis représenter sur une figure en 4 tableaux, le signal initial $yi$, le signal bruité $yb$, le signal lissé $y\ell$ et la superposition de $yi$ et $y\ell$. Test `SP3(1000,8)`, utiliser `subplot`.



Tester aussi `SP3(1000,5)`. Expliquer la différence. Mais où est passé le bruit dans le premier cas ?

**4.** Cette fois-ci le signal périodique n'est plus impair. On utilise la même méthode avec les $2M+1$ fonctions

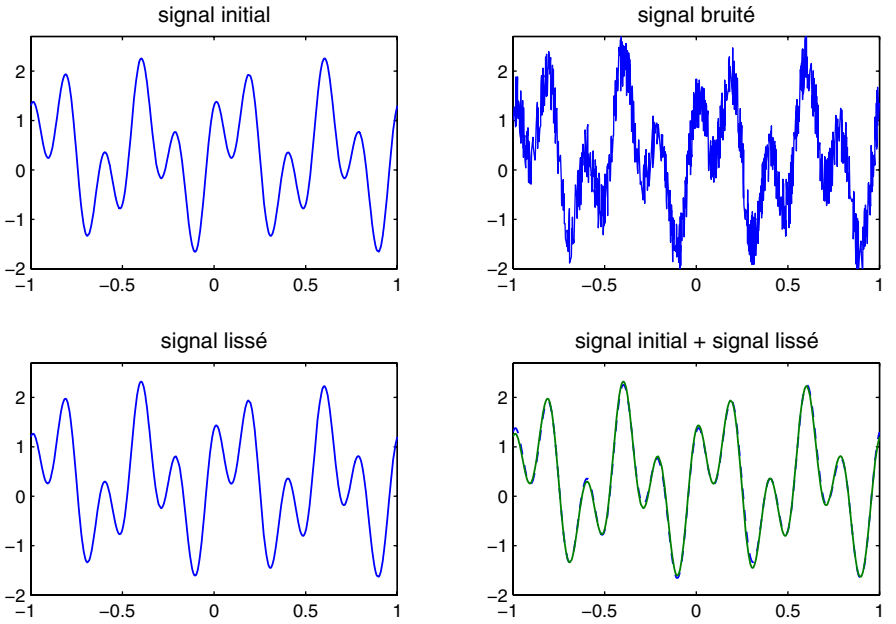
$$1 = \cos 0\pi t,\ \sin 1\pi t,\ \cos 1\pi t,\ \sin 2\pi t,\ \cos 2\pi t,\ \ldots,\ \sin M\pi t,\ \cos M\pi t.$$

et `signal2.m`

```
function [x,y,yb]=signal2(N);
x=sort(2*rand(N,1)-1);
x(1)=-1;x(N)=1;
y=0.3+cos(10*pi*x)+sin(4*pi*x);
yb=y+(2*rand(N,1)-1)*0.6;
```

En appelant la fonction `signal2` construire une 4ème fonction `SP4.m` qui entre $M$ et $N$, calcule le signal lissé et fait les mêmes dessins que dans la question précédente. On affichera $U$ qui a $2M+1$ composantes. Test `SP4(1000,11)`.

```
>> SP4(1000,11)
U =
    0.2944
    0.0104
   -0.0094
   -0.0023
   -0.0063
   -0.0097
   -0.0218
    1.0357
    0.0056
    0.0076
   -0.0148
   -0.0131
   -0.0094
   -0.0052
    0.0216
    0.0259
   -0.0124
    0.0064
    0.0025
    0.0143
    1.0249
   -0.0424
   -0.0098
```




## 6.4 Méthode du filtre de Savitsky-Golay

Soient $(x_i, f_i)_{i=1\ldots n}$, $n$ valeurs mesurées en les abscisses $x_i$ supposées équidistantes, $x_{i+1} - x_i = \Delta x$. On cherche à lisser ces valeurs supposées bruitées. Pour un indice $i$ donné, l'idée est de remplacer $f_i$ par une valeur $g_i$ approchant $f_i$ par une moyenne des valeurs aux points voisins.

Soient $N_L$ et $N_R$ deux entiers strictement positifs et $M$ un entier positif. Pour $i$ fixé vérifiant $N_L + 1 \leqslant i \leqslant n - N_R$, au point $x_i$, nous utilisons les points voisins d'abscisses $x_{i-N_L}, \ldots, x_i, \ldots, x_{i+N_R}$. $p_i$ est le polynôme de degré $M$ lissant ces $N_L + N_R + 1$ points, c'est-à-dire que $p_i$ réalise le minimum de $J(p) = \sum_{j=i-N_L}^{i+N_R} (p(x_j) - f_j)^2$, $p \in \mathbb{P}_M$. On définit ensuite $g_i = p_i(x_i)$.

On voit que le choix des données porte sur $N_L$ et $N_R$, respectivement nombre de points à gauche et à droite et sur le degré du polynôme de lissage.

Le choix de la base $\{1, \dfrac{(x-x_i)^1}{(\Delta x)^1}, \ldots, \dfrac{(x-x_i)^M}{(\Delta x)^M}\}$ de $\mathbb{P}_M$ est la clé de la méthode. Attention, cette base est de cardinal $M + 1$. Si $p \in \mathbb{P}_M$, on peut donc écrire

$$p(x) = \sum_{k=0}^{M} \alpha_k \frac{(x-x_i)^k}{(\Delta x)^k}.$$

où $\alpha = \begin{pmatrix} \alpha_M \\ \vdots \\ \alpha_0 \end{pmatrix}$ désigne le vecteur des coefficients de $p$. On vérifie que $p(x_i) = e_{M+1}^T \times \alpha$ où $e_{M+1} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix}$.

Pour résoudre le problème, $\alpha$ est déterminé en minimisant l'écart

$$J(\alpha) = \sum_{j=i-N_L}^{i+N_R} (p(x_j) - f_j)^2, \, p \in \mathbb{P}_M.$$

**1.** Montrer que le vecteur $\begin{pmatrix} p(x_{i-N_L}) \\ \vdots \\ p(x_i) \\ \vdots \\ p(x_{i+N_R}) \end{pmatrix}$ s'écrit sous la forme $A\alpha$ où $A$ est une matrice de $\mathbb{R}^{(N_L+N_R+1)\times(M+1)}$ à préciser.

**2.** Si $F_i = (f_{i-N_L}, ..., f_i, ..., f_{i+N_R})^T$, montrer que $J(\alpha) = \|A\alpha - F_i\|_2^2$.

**3.** En déduire un système linéaire carré vérifié par $\bar{\alpha}$ minimisant $J(\alpha)$.

**4.** En déduire $\bar{\alpha}$.

**5.** Construire $g_i = p_i(x_i)$ en fonction de $e_{M+1}$, $A$ et $F_i$.

**6.** En déduire une expression de $g_i$ sous la forme $C^T \times F_i$.

**7.** $C$ dépend-il de $i$ ?

**8.** **Programmation :** Construire la fonction `signal1.m` échantillonnant la fonction

$$F(x) = \frac{1}{1+100(x-0.2)^2} + \frac{1}{1+500(x-0.4)^2)} + \frac{1}{1+1\,000(x-0.6)^2} + \frac{1}{1+10\,000(x-0.8)^2}$$

en $n$ points équidistants sur $[0, 1]$. Test avec $n = 6$ sortie de $x$ et $y$, abscisses et ordonnées.

```
>> [x,y]=signal1(6)
x =
         0    0.2000    0.4000    0.6000    0.8000    1.0000

y =
    0.2153    1.0541    1.2250    1.1089    1.0638    0.0296
```

**9.** Bruiter ce signal en additionnant au vecteur $y$ le vecteur $\frac{10}{100}V$ où $V$ est un vecteur aléatoire à $n$ composantes entre $-1$ et 1. On pourra utiliser la fonction `rand.m` de `Matlab`. Sauvegarde sous `signal2.m`. Test avec $n = 6$.

```
>> [x,yi,yb]=signal2(6)
x =
         0    0.2000    0.4000    0.6000    0.8000    1.0000

yi =
    0.2153    1.0541    1.2250    1.1089    1.0638    0.0296

yb =
    0.2509    1.0239    1.1572    1.1423    1.0182    0.0519
```

Comme on a introduit un facteur aléatoire, les valeurs de *yb* peuvent être différentes.

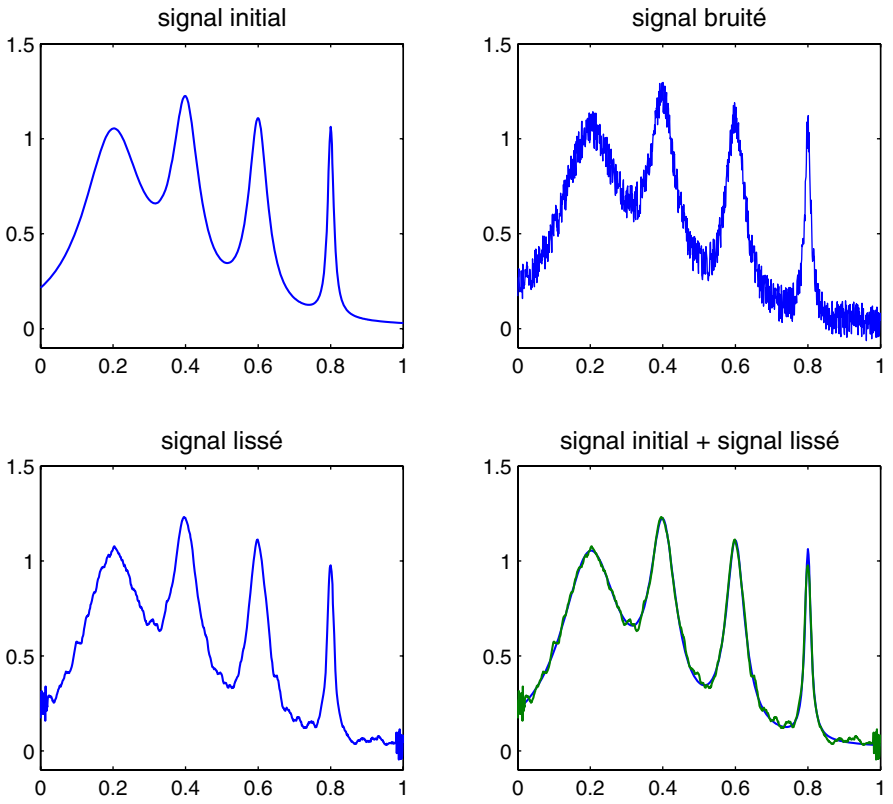
**10.** Construire la matrice $A$. Sauvegarde sous `gol1.m`. Test avec $M = 4$, $N_L = 4$, $N_R = 5$.

```
>> gol1
A =
   256   -64    16    -4     1
    81   -27     9    -3     1
    16    -8     4    -2     1
     1    -1     1    -1     1
     0     0     0     0     1
     1     1     1     1     1
    16     8     4     2     1
    81    27     9     3     1
   256    64    16     4     1
   625   125    25     5     1
```

**11.** Compléter le programme pour calculer $C$. Sauvegarde sous `gol2.m`. Même test.

```
>> gol2
C =
   -0.0000
   -0.0583
    0.0874
    0.2622
    0.3543
    0.3147
    0.1573
   -0.0408
   -0.1399
    0.0629
```

**12.** Compléter en faisant appel à `signal 2.m` pour dessiner en 4 figures : signal initial, signal bruité, signal lissé et la superposition du signal initial et du signal lissé. Sauvegarde sous `gol3.m`. Test avec $n = 1000$, $M = 4$, $N_L = N_R = 20$. Puis tester plusieurs valeurs de $M$, $N_L$ et $N_R$.




## 6.5 Équation différentielle et moindres carrés

Soit $f \in \mathcal{C}^0([0,1])$. On considère le problème suivant :

Trouver $u(x)$ tel que

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

et on admet que ce problème a une solution unique.

On cherche une approximation de la solution $u$ sous la forme d'un polynôme

$$p(x) = \sum_{j=1}^{M} \alpha_j w_j(x) \text{ où } w_j(x) = x^{j-1}.$$

Pour cela, on discrétise ]0, 1[ par $x_i = ih$ avec $h = \dfrac{1}{n+1}$ et $i = 1, \ldots, n$ et on minimise

$$J(p) = p(0)^2 + p(1)^2 + \sum_{i=1}^{n} \big( - p''(x_i) + p(x_i) - f(x_i) \big)^2 \tag{6.1}$$

On a
$$\begin{aligned} p(0) &= \sum_{j=1}^{M} \alpha_j \, w_j(0) = \alpha_1 \\ p(1) &= \sum_{j=1}^{M} \alpha_j \\ p''(x_i) &= \sum_{j=3}^{M} \alpha_j (j-1)(j-2) x_i^{j-3} \\ p(x_i) &= \sum_{j=1}^{M} \alpha_j x_i^{j-1} \end{aligned}$$

On construit la matrice $A$ pour que le problème (6.1) se ramène à : $\min_{\alpha} \|A\alpha - b\|^2$. Donc

$$A = \begin{pmatrix} 1 & 0 & \ldots & & \ldots & 0 \\ \vdots & & & & & \\ \vdots & & & & & \\ \vdots & & & -(j-1)(j-2)x_i^{j-3} + x_i^{j-1} & & \\ \vdots & & & & & \\ 1 & 1 & \ldots & & \ldots & 1 \end{pmatrix} \quad , \; b = \begin{pmatrix} 0 \\ f(x_1) \\ \vdots \\ \vdots \\ f(x_n) \\ 0 \end{pmatrix}$$

$M$ colonnes

Attention aux décalages d'indice, `Matlab` n'acceptant pas les indices 0.

**1.** Créer un tableau d'abscisses $x_i = ih$ pour $1 \leqslant i \leqslant n$. Puis construire la matrice $A$. Sauvegarder sous `EQD1.m`. Tester avec $M = 4, n = 6$. Afficher $A$.

```
>> EQD1
A =
    1.0000         0         0         0
    1.0000    0.1429   -1.9796   -0.8542
    1.0000    0.2857   -1.9184   -1.6910
    1.0000    0.4286   -1.8163   -2.4927
    1.0000    0.5714   -1.6735   -3.2420
    1.0000    0.7143   -1.4898   -3.9213
    1.0000    0.8571   -1.2653   -4.5131
    1.0000    1.0000    1.0000    1.0000
```

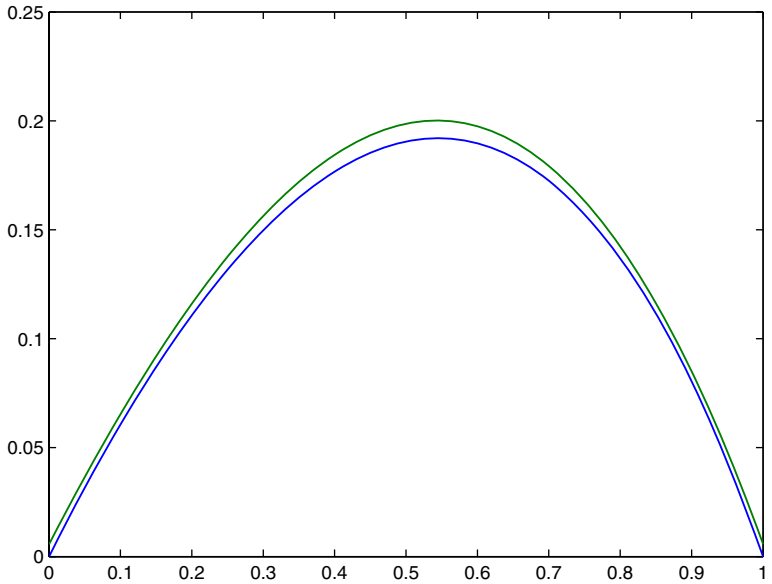
**2.** Construire le second membre $b$. Pour cela créer un fichier function `f.m`. Afficher $b$. Sauvegarder sous `EQD2.m`. Tester avec $M = 4, n = 6$ et $f(x) = \exp(x)$.

```
>> EQD2
b =
         0
    1.1536
    1.3307
    1.5351
    1.7708
    2.0427
    2.3564
         0
```

**3.** Calculer les coefficients $\alpha$ en résolvant le système $A^T A\alpha = A^T b$. Sauvegarder sous `EQD3.m`. Tester avec $M = 4, n = 6$.

```
>> EQD3
alpha =
    0.0056
    0.6355
   -0.3626
   -0.2728
```

**4.** Calculer et dessiner le polynôme $p$ sur l'intervalle $[0, 1]$. Comparer sur un graphe à la solution exacte $u(x) = \dfrac{e}{2(e - 1/e)}(e^x - e^{-x}) - \dfrac{x}{2}e^x$. Sauvegarder sous `EQD4.m`. Tester avec $M = 4, n = 6$.



**5.** Calculer $erreur = \|p - \text{solexact}\|_\infty$. En fait on calculera $\max_{k=0,\ldots,1000} |p(t_k) - \text{solexact}(t_k)|$ où $t_k = k/1000$. Sauvegarder sous `EQD5.m`. Tester avec $M = 4$, $n = 6$.

```
>> EQD5
erreur =
    0.0082
```

**6.** Étudier l'erreur lorsque $M$ varie. On prendra $n = M + 2$ et on créera un tableau de valeurs de $M$ par exemple $tM = [10, 15, 20, 35, 50, 75]$. Sauvegarde sous `EQD6.m`. Que remarque-t-on et pourquoi ?

```
>> EQD6
tM =
    10    15    20    35    50    75

taberreur =
  1.0e-009 *
    0.0434    0.4952    0.2172    0.0631    0.3787    0.1243
```

**7.** Construire un jeu d'essais tel que la solution exacte soit la fonction sinusoïdale $u(x) = \sin(\pi x)$. Sauvegarder sous `EQD7.m` en construisant une fonction `f1.m`. Tracer pour plusieurs valeurs de $M$, solutions exacte et approchée.

## DU MAL À DÉMARRER ?

### 6.1 Mise en équation

**2.** Rappel Si $U \in \mathbb{R}^n$, $\|U\|_2 = \left(\sum_{i=1}^{n} u_i^2\right)^{1/2}$.

**3.** Théorème de Pythagore : $\|u - b\|^2 = \|u - \bar{u}\|^2 + \|\bar{u} - b\|^2$.

**4.** $\text{Im}A = \{V \in \mathbb{R}^n / \exists U \in \mathbb{R}^m,\ V = AU\}$.

### 6.2 Droite et parabole de régression

Construire la matrice $A$.

### 6.3 Signal périodique

**1.** Pour construire $A = (\sin(j\pi x_i)$, on peut commencer par $B = (jx_i)$.

**4.** Pour construire $A$, il s'agit de gérer un tableau, colonne par colonne ; on peut revoir le chapitre 1.

### 6.4 Filtre de Savitsky-Golay

**1.** Étudier la ligne $j$ i.e. $p(x_j)$.

**2.** Si $U = (u_{i-N_L}, \ldots, u_i, \ldots, u_{i+N_R})^T$ alors $\|U\|_2 = \left( \sum_{j=i-N_L}^{i+N_R} u_j^2 \right)^{1/2}$.

## CORRIGÉS DES EXERCICES

### 6.1 Mise en équation

Sachant que $A = (a_{ij})_{i=1,\ldots,n,\ j=1,\ldots,m}$ avec $a_{ij} = \varphi_j(x_i)$, on en déduit que pour $\alpha \in \mathbb{R}^n$,

$$A\alpha = \begin{pmatrix} \sum_{j=1}^{m} \alpha_j \varphi_j(x_1) \\ \vdots \\ \sum_{j=1}^{m} \alpha_j \varphi_j(x_n) \end{pmatrix} = \begin{pmatrix} \varphi(x_1) \\ \vdots \\ \varphi(x_n) \end{pmatrix}, \text{ si bien que } \|A\alpha - y\|_2 = \left( \sum_{i=1}^{n} (\varphi(x_i) - y_i)^2 \right)^{1/2} = J(\alpha).$$

Soit $\bar{u}$ la projection orthogonale de $y$ sur $E$. Le vecteur $\bar{u} - y$ est donc orthogonal à $E$. Si $u$ est un vecteur quelconque de $E$, $u - y = u - \bar{u} + \bar{u} - y$ et

$$\begin{aligned} \|u - y\|_2^2 &= \|u - \bar{u}\|_2^2 + \|\bar{u} - y\|_2^2 + 2(u - \bar{u}).(\bar{u} - y) \\ &= \|u - \bar{u}\|_2^2 + \|\bar{u} - y\|_2^2 \end{aligned}$$

puisque $\bar{u} - y \perp u - \bar{u}$. On voit donc que le minimum de $\|u - y\|_2^2$ ou de $\|u - y\|_2$ est atteint quand $u = \bar{u}$.

Im$A$ est un sous espace vectoriel de $\mathbb{R}^n$ donc le minimum de $\|A\alpha - y\|_2$ est obtenu en $\bar{u} = A\bar{\alpha}$, projection orthogonale de $y$ sur Im$A$.

**Attention**, la projection orthogonale $\bar{u}$ est unique, mais $\bar{\alpha}$ n'a aucune raison d'être unique. Ainsi si les $\varphi_j$ sont définies par $\varphi_j(x) = \sin j\pi x$ et que les $x_i$ sont des entiers alors $A$ est la matrice nulle et tout $\alpha$ de $\mathbb{R}^m$ réalise $\bar{u} = A\alpha$. L'unicité de $\bar{\alpha}$ est obtenue si $A$ est injective autrement dit si le rang de la matrice $A$ est $m$. Dans ce cas, pour tout $v = A\beta$ de Im$A$, on a $v^T(A\bar{\alpha} - y) = 0$ soit pour tout $\beta$ de $\mathbb{R}^m$, $\beta^T A^T(A\bar{\alpha} - y) = 0$ ; on en déduit que $A^T(A\bar{\alpha} - y) = 0$ ou $A^T A\bar{\alpha} = A^T y$.

### 6.2 Droite et parabole de régression

Les coefficients $\alpha_i$ sont solutions de $A^T A\alpha = A^T y$ où

$$A = \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ 1 & x_3 \\ 1 & x_4 \\ 1 & x_5 \end{pmatrix} \text{ pour la droite et } A = \begin{pmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ 1 & x_3 & x_3^2 \\ 1 & x_4 & x_4^2 \\ 1 & x_5 & x_4^2 \end{pmatrix} \text{ pour la parabole.}$$

Pour la droite le système est $\begin{pmatrix} 5 & 0 \\ 0 & 10 \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \alpha_2 \end{pmatrix} = \begin{pmatrix} 15 \\ 20 \end{pmatrix}$ qui donne $\alpha_1 = 2$ et $\alpha_2 = 3$. Pour la parabole le système est $\begin{pmatrix} 5 & 0 & 10 \\ 0 & 10 & 0 \\ 10 & 0 & 34 \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \end{pmatrix} = \begin{pmatrix} 15 \\ 20 \\ 44 \end{pmatrix}$ qui donne $\alpha_1 = 1$, $\alpha_2 = 2$ et $\alpha_3 = 1$.

```
>> x=[-2 -1 0 1 2];y=[3/2 -1 1 5 17/2];
>> a=polyfit([-2 -1 0 1 2],[3/2 -1 1 5 17/2],1)
a =
    2.0000    3.0000
>> a=polyfit([-2 -1 0 1 2],[3/2 -1 1 5 17/2],2)
a =
    1.0000    2.0000    1.0000
>> xx=-2:0.01:2;
subplot(1,2,1)
plot(x,y,'bo',xx,2*xx+3,'b')
subplot(1,2,2)
plot(x,y,'bo',xx,(xx+1).^2,'b')
```

## 6.3 Signal périodique

SP3.m

```
function SP3(N,M);
[x,yi,yb]=signal1(N);
A=sin(x*(1:M)*pi);
U=(A'*A)\ (A'*yb)
yl=A*U;
subplot(2,2,1)
plot(x,yi)
axis([-1 1 -1.6 1.6])
title('signal␣initial')
subplot(2,2,2)
plot(x,yb)
axis([-1 1 -1.6 1.6])
title('signal␣bruité')
subplot(2,2,3)
plot(x,yl)
axis([-1 1 -1.6 1.6])
title('signal␣lissé')
subplot(2,2,4)
plot(x,yi,x,yl)
axis([-1 1 -1.6 1.6])
title('signal␣initial␣+␣signal␣lissé')
```

### Ce qu'il faut retenir de cet exercice

Construction de la matrice $(jx_i)$ sans boucle avec l'instruction `x*(1 :M)`.

Le bruit est essentiellement dans l'orthogonal s'il existe suffisamment de fonctions de base, ainsi : $yb \simeq y\ell + bruit$. Sinon, le bruit reste dans $y\ell$.

`SP4.m`

```
function SP4(N,M);
[x,yi,yb]=signal2(N);
A1=cos(x*(0:M)*pi);
A2=sin(x*(1:M)*pi);
A(:,1:2:2*M+1)=A1;
A(:,2:2:2*M)=A2;
U=(A'*A)\ (A'*yb)
yl=A*U;
subplot(2,2,1)
plot(x,yi)
axis([-1 1 -2.1 2.3])
title('signal␣initial')
subplot(2,2,2)
plot(x,yb)
axis([-1 1 -2.3 2.3])
title('signal␣bruité')
subplot(2,2,3)
plot(x,yl)
axis([-1 1 -2.3 2.3])
title('signal␣lissé')
subplot(2,2,4)
plot(x,yi,x,yl)
axis([-1 1 -2.3 2.3])
title('signal␣initial␣+␣signal␣lissé')
```

**Ce qu'il faut retenir de cet exercice**

À nouveau, construction de la matrice $A$ sans boucle. L'instruction `A( :,1 :2 :2*M+1)` permet de retenir une colonne sur deux.

## 6.4 Filtre de Savistky-Golay

**1.** Si $p(x) = \sum_{k=0}^{M} \alpha_k \frac{(x - x_i)^k}{(\Delta x)^k}$, pour écrire le vecteur $U = (p(x_{i-N_L}), \ldots, p(x_{i+N_R})^T$ sous la forme $A\alpha$, en numérotant les $N_L + N_R + 1$ lignes de $A = (a_{jk})$ (et de $U$) de $i - N_L$ à $i + N_R$, la $j$ème ligne $p(x_j)$ correspond à la $j$ème ligne de $A$ multipliée par le vecteur $\alpha$ soit

$$p(x_j) = \sum_{k=0}^{M} \alpha_k \frac{(x_j - x_i)^k}{(\Delta x)^k} = \sum_{k=0}^{M} a_{jk}\alpha_k.$$

On voit donc qu'on peut choisir $a_{jk} = \frac{(x_j - x_i)^k}{(\Delta x)^k}$. On remarque alors que $x_j - x_i = (j - i)\Delta x$, ce qui simplifie le coefficient. En faisant varier $j$ de $i - N_L$ à $i + N_R$, on obtient la matrice $A$ à

$N_L + N_R + 1$ lignes et $M + 1$ colonnes :

$$A = \begin{pmatrix} (-N_L)^M & (-N_L)^{M-1} & \dots & -N_L & 1 \\ (-N_L+1)^M & & \dots & -N_L+1 & 1 \\ \vdots & & & \vdots & \\ (-1)^M & (-1)^{M-1} & \dots & -1 & 1 \\ 0 & 0 & \dots & 0 & 1 \\ 1 & 1 & \dots & 1 & 1 \\ \vdots & & & \vdots & \\ N_R^M & N_R^{M-1} & \dots & N_R & 1 \end{pmatrix}.$$

Ce qui est remarquable, c'est que $A$ ne dépend pas du point $x_i$.

**2.** Si pour $F = (f_{i-N_L}, \dots, f_{i+N_R})^T$, nous numérotons à nouveau les lignes de $i - N_L$ à $i + N_R$, alors la $j$ème ligne de $A\alpha - F_i$ est exactement $p(x_j) - f_j$ et donc

$$J(\alpha) = \sum_{j=i-N_L}^{i+N_R} (p(x_j) - f_j)^2 = \|A\alpha - F_i\|_2^2$$

en reprenant la définition de la norme 2 (cf. Chapitre 3).

**3.** On se retrouve dans le cadre du paragraphe sur la mise en équation et le minimum de $J$ est atteint en $\bar{\alpha}$ où $A\bar{\alpha}$ est la projection orthogonale de $F_i$ sur $\mathrm{Im}A$ soit $A^T A\bar{\alpha} = A^T F_i$.

**4.** En supposant que $N_L + N_R \geqslant M$, en extrayant les $M + 1$ premières lignes de $A$, on a une matrice de Vandermonde qui est de rang $M + 1$. Le système précédent admet une solution unique $\bar{\alpha} = (A^T A)^{-1} A^T F_i$.

**5.**, **6.** et **7.** La solution cherchée vérifie $g_i = p(x_i) = e_{M+1}^T (A^T A)^{-1} A^T F_i = C^T F_i$ où $C = A(A^T A)^{-1} e_{M+1}$. On voir clairement que comme $A$, $C$ ne dépend pas de $i$ et n'est à calculer qu'une fois.

Il reste une contrainte, c'est que $i$ doit varier entre $N_L + 1$ et $n - N_R$ ; c'est dire qu'il est impossible de filtrer les premiers et derniers éléments par cette méthode.

**8...12** `gol123.m`

```
clear
NL=20;NR=20;
M=4;
% calcul de A
A=ones(NL+NR+1,M+1);
V=(-NL:NR)';
A(:,M)=V;
for j=M-1:-1:1
    A(:,j)=A(:,j+1).*V;
end;
% calcul de C
```

```
e=zeros(M+1,1);
e(M+1)=1;
C=A*(A'*A)^(-1)*e;
n=1000;
[x,yi,yb]=signal2(n);
% Filtrage
yl=yb;
for i=NL+1:n-NR
    yl(i)=C'*yb(i-NL:i+NR)';
end;
% Graphes
subplot(2,2,1)
plot(x,yi)
axis([0 1 -0.1 1.5])
title('signal␣initial')
subplot(2,2,2)
plot(x,yb)
axis([0 1 -0.1 1.5])
title('signal␣bruite')
subplot(2,2,3)
plot(x,yl)
axis([0 1 -0.1 1.5])
title('signal␣lisse')
subplot(2,2,4)
plot(x,yi,x,yl)
axis([0 1 -0.1 1.5])
title('signal␣initial␣+␣signal␣lisse')
```

**Ce qu'il faut retenir de cet exercice**

Cette fois-ci une boucle est utilisée pour construire $A$. Elle évite l'instruction puissance.

## 6.5 Équation différentielle

EQD1234.m

```
M=4;
n=6;
h=1/(n+1);
x=(h:h:1-h)';
A1=x*ones(1,M);
for j=1:M
    A2(:,j)=-(j-1)*(j-2)*A1(:,j).^(j-3)+A1(:,j).^(j-1);
end
A=[zeros(1,M);A2;ones(1,M)];
A(1,1)=1;
b=[0;f(x);0];
alpha=(A'*A)\ (A'*b);
t=0:1/1000:1;
e=exp(1);
uex=e/(2*(e-1/e))*(exp(t)-exp(-t))-t/2.*exp(t);
uapp=polyval(alpha(M:-1:1),t);
plot(t,uex,t,uapp)
erreur=norm(uex-uapp,inf)
```

### Ce qu'il faut retenir de cet exercice

Pour calculer $te^t$, où $t$ est un tableau, l'instruction Matlab est `t.*exp(t)`. Sur la différence entre * et .* revoir le chapitre 1.

EQD6.m

```
clear
tM=[10 15 20 35 50 75];
for i=1:length(tM)
M=tM(i)
n=M+2;
h=1/(n+1);
x=(h:h:1-h)';
A1=x*ones(1,M);
for j=1:M
A2(:,j)=-(j-1)*(j-2)*A1(:,j).^(j-3)+A1(:,j).^(j-1);
end
A=[zeros(1,M);A2;ones(1,M)];
A(1,1)=1;
clear A2
b=[0;f(x);0];
alpha=(A'*A)\ (A'*b);
t=0:1/1000:1;
e=exp(1);
uex=e/(2*(e-1/e))*(exp(t)-exp(-t))-t/2.*exp(t);
uapp=polyval(alpha(M:-1:1),t);
    %plot(t,uex,t,uapp)
taberreur(i)=norm(uex-uapp,inf);
end;
tM
taberreur
```

On remarque que l'erreur diminue très vite puis qu'elle reste stable, voire en augmentation. En effet, la matrice $A^TA$ est très mal conditionnée (voir le chapitre 3) quand $M$ devient grand. On peut alors s'interroger sur le choix d'une base polynômiale plus adaptée.

Pour conclure par un autre exemple, on prendra $f1(x) = (\pi^2 + 1)\sin(\pi x)$.

# 7 Courbes de Bézier

Ce chapitre propose une introduction aux nouveaux outils mathématiques créés dans les années 60 pour la CAO (Conception assistée par ordinateur), CFAO (Conception Fabrication...), DAO (Dessin...). L'objectif mathématique est de définir et de tracer des courbes (ou des surfaces) à partir d'une forme donnée. Bien entendu ces outils ont progressé et aujourd'hui dans les logiciels, les polynômes de Bernstein ont été remplacés par d'autres fonctions : B-splines ou fonctions rationnelles. Les utilisations sont nombreuses en aéronautique, pour l'imagerie médicale... mais aussi pour la fabrication de dessins animés et de jeux vidéo.

Le rappel de cours est remplacé par deux exercices qui introduisent les notions de polygone de contrôle puis de courbe de Bézier, sous forme de fonction paramétrique définie par récurrence puis réécrite sous forme explicite dans la base de Bernstein. Les deux exercices suivants permettent d'aborder des contraintes de construction : comment « recoller » deux courbes, comment imposer des formes données. Enfin le dernier permet d'écrire une fonction paramétrique polynômiale dans la base de Bernstein et de déterminer ainsi le polygone de contrôle correspondant.

## ÉNONCÉS DES EXERCICES

### 7.1 Algorithme de de Casteljau

On se place dans l'espace affine $E = \mathbb{R}^d$ avec $d = 2$ ou $d = 3$. A partir de $n+1$ points $b_i$, $i = 0, \ldots, n$ de $E$, pour $t \in [0,1]$, on construit par récurrence les courbes paramétrées suivantes :

$$\begin{array}{rll} \text{initialisation :} & b_i^0(t) & = b_i, \text{ pour } i = 0, \ldots, n \\ \text{pour } r = 1, \ldots, n, & b_i^r(t) & = (1-t)\, b_i^{r-1}(t) + t\, b_{i+1}^{r-1}(t), \text{ pour } i = 0, \ldots, n-r. \end{array}$$

C'est l'algorithme de de Casteljau ; le polygone $\{b_0, \ldots, b_n\}$ s'appelle polygone de contrôle de la courbe de Bézier $b_0^n(t)$ aussi notée $b^n(t)$ ou $\mathcal{B}(b_0, \ldots, b_n; t)$.

**1.** Calculer $b^n(0)$ et $b^n(1)$.

**2.** Soit $\varphi$ une transformation affine de $E$. Montrer que l'image de la courbe de Bézier par $\varphi$ est encore une courbe de Bézier ; plus précisément, montrer que

$$\varphi\left(\mathcal{B}(b_0, \ldots, b_n; t)\right) = \mathcal{B}\left(\varphi(b_0), \ldots, \varphi(b_n); t\right).$$

### ➤ Programmation

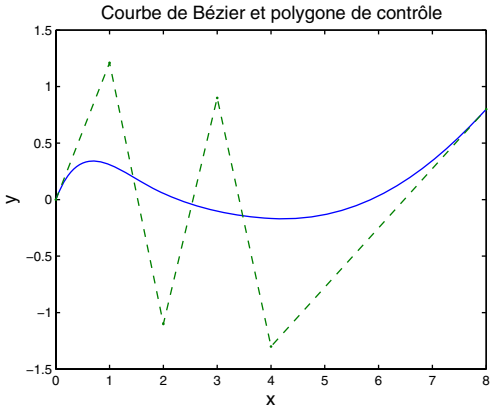
On suppose que le polygone de contrôle est connu sous forme de tableau $pc$ à $N = n+1$ colonnes et 2 ou 3 lignes (abscisse, ordonnée, hauteur) ; le nombre de lignes sera appelé $dim$. Il s'agit de déterminer la courbe de Bézier en $m$ points $t$ équirépartis sur $[0, 1]$ puis de la tracer ainsi que le polygone de contrôle. Pour cela, à chaque étape $r$, on construit un tableau à 3 dimensions $b(1 : dim, 1 : m, 1 : n - r + 1)$. Pour programmer l'algorithme, on pourra aussi utiliser un tableau $tt$ qui reproduit le vecteur $t$ sur les $dim$ lignes permettant ainsi des multiplications composante par composante avec des sous tableaux de $b$.

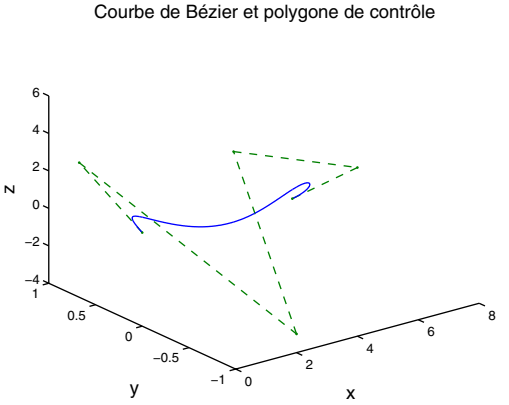
**1.** Écrire une fonction `C=c-bezier(pc,m)` qui étant donné un polygone de contrôle $pc$ calcule la fonction paramétrée $b^n(t)$ en $m$ points équirépartis sur $[0, 1]$. Il faudra évidemment déterminer $n$ et $dim$. Test avec $m = 6$ et $[0, 1, 2, 3; 0, 1, -1, 1]$.

```
>> pc=[0,1,2,3;0,1,-1,1];m=6;C=c_bezier(pc,m)
C =
         0    0.6000    1.2000    1.8000    2.4000    3.0000
         0    0.2960    0.2080    0.0720    0.2240    1.0000
```

Noter que, compte tenu de la répartition uniforme des abscisses, la première composante de $b^n(t)$ est $x(t) = 3t$.

**2.** Écrire un programme permettant le tracé du polygone de contrôle et de la courbe en $m$ points. Utiliser les fonctions `Matlab : plot, plot3` en fonction du nombre de composantes $dim$. Sauvegarde dans `trace.m`. Test avec $m = 500$ et $pc = [0, 1, 2, 3, 4, 8; 0, 1.21, -1.1, 0.9, -1.3, 0.8]$, $pc = [0, 1, 2, 3, 4, 8; 0, 1, -1, 0, -1, 1; 1, 2, -3, 4, 5, -3]$, successivement.



## 7.2 Polynômes de Bernstein

Pour $n$ entier positif et $\binom{n}{i} = \dfrac{n!}{(n-i)!i!}$, on définit les polynômes de Bernstein

$$B_i^n(t) = \binom{n}{i} t^i (1-t)^{n-i}, \text{ pour } i = 0, \ldots, n$$
$$B_i^n(t) = 0, \text{ pour } i \notin \{0, \ldots, n\}$$

**1.** Montrer que les $B_i^n,\ i = 0, \ldots, n$ forment une base de $\mathbb{P}_n$, puis que

$$B_i^n(t) = (1-t)B_i^{n-1}(t) + t\,B_{i-1}^{n-1}(t),\ n \geqslant 1,\ i \in \mathbb{Z}. \tag{7.1}$$

**2.** Montrer que

(a) $B_i^n(t) \geqslant 0$ pour $t \in [0,1]$,

(b) $B_i^n(t) = B_{n-i}^n(1-t)$,

(c) $\displaystyle\sum_{i=0}^{n} B_i^n(t) = 1$ et $\displaystyle\sum_{i=0}^{n} \frac{i}{n} B_i^n(t) = t$.

Soit $\{b_0, \ldots, b_n\}$ un polygone de $E$. Nous allons établir le lien entre la courbe de Bézier et les polynômes de Bernstein.

**3.** Montrer que pour $t \in [0,1]$ et pour $r = 0, \ldots, n$, $b_i^r(t) = \displaystyle\sum_{j=0}^{r} b_{i+j} B_j^r(t),\ i = 0, \ldots, n-r$

en particulier :

$$b^n(t) = b_0^n(t) = \sum_{j=0}^{n} b_j B_j^n(t). \tag{7.2}$$

**4.** Montrer que $\dfrac{dB_i^n}{dt}(t) = n\left(B_{i-1}^{n-1}(t) - B_i^{n-1}(t)\right)$ pour $n \geqslant 1$ et $i \in \mathbb{Z}$. En déduire $\dfrac{db^n}{dt}(t)$ en fonction des $\Delta b_i = b_{i+1} - b_i$, puis déterminer $\dfrac{d^p b^n}{dt^p}(t)$. On montrera :

$$\frac{db^n}{dt}(t) = \sum_{k=0}^{n-1} \Delta b_k B_k^{n-1}(t) \text{ avec } \Delta b_k = b_{k+1} - b_k. \tag{7.3}$$

$$\frac{d^p b^n}{dt^p}(t) = \frac{n!}{(n-p)!} \sum_{k=0}^{n-p} \Delta^p b_k B_k^{n-p}(t) \text{ où } \Delta^p b_k = \Delta^{p-1} b_{k+1} - \Delta^{p-1} b_k. \tag{7.4}$$

Enfin, citons le théorème de Weierstrass dont une élégante démonstration est donnée dans [28]
Théorème : *Soit $f$ une fonction continue de $[a,b]$ dans $E=\mathbb{R}^d$. On définit la suite de polynômes* $f_n(x)=\sum_{i=0}^{n} f(i/n)B_i^n(t)$ *où* $t=\dfrac{x-a}{b-a}$. *Alors la suite $(f_n)$ converge uniformément vers $f$ i.e.*

$$\lim_{n\to+\infty}\sup_{x\in[a,b]}|f(x)-f_n(x)|=0.$$

## 7.3 Raccords entre des courbes

On se place dans un repère orthonormé du plan et l'on considère le polygone de contrôle formé des points suivants : $a_0(-1,0)$, $a_1(-1,1)$, $a_2(1,0)$, $a_3(1,1)$. On construit la courbe de Bézier correspondante $P_3(t)=\mathcal{B}(a_0,a_1,a_2,a_3,t)$ pour $t\in[0,1]$.

**1.** Déterminer le polygone de contrôle de $P_3'(t)$ puis celui de $P_3''(t)$. Préciser l'allure du graphe de $P_3''(t)$.

**2.** On veut prolonger la courbe $P_3$ avec un raccord $\mathcal{C}^1$ en une courbe de Bézier : $Q_2(t)=\mathcal{B}(b_0,b_1,b_2,t)$ où $b_0=a_3$ et $b_2(3,0)$. Déterminer les coordonnées de $b_1$.

**3.** On veut cette fois un prolongement $\mathcal{C}^2$ de l'autre côté de la courbe initiale par une courbe de Bézier : $R_3(t)=\mathcal{B}(c_0,c_1,c_2,c_3,t)$ où $c_0(3,0)=b_2$ et $c_3=a_0$. Déterminer les coordonnées de $c_1$ et $c_2$.

**4.** Construire tous les points $a_i$, $b_j$, $c_k$ ainsi que les 3 courbes $P_3$, $Q_2$, $R_3$.

## 7.4 Contraintes de formes

Pour $t\in[0,1]$, on considère les 2 courbes paramétrées d'équations

$$P_1(t)=\begin{cases} x(t)=t \\ y(t)=1-3t+9t^3 \end{cases},\quad Q_1(t)=\begin{cases} u(t)=t-1 \\ v(t)=2-3t+6t^2-4t^3 \end{cases}.$$

**1.** Étudier les variations de la fonction $y$ et montrer que cette fonction est strictement positive pour $t\in[0,1]$.

**2.** Déterminer le polygone de contrôle $\{b_0,b_1,b_2,b_3\}$ de la courbe $P_1(t)$ interprétée comme une courbe de Bézier. Si $b_i=\begin{pmatrix}x_i\\y_i\end{pmatrix}$, on rappelle que $x_0=0, x_1=1/3, x_2=2/3, x_3=1$. Pour déterminer les $y_i$, on pourra calculer les valeurs de $y$ et $y'$ en 0 et 1.

**3.** Dessiner courbe et polygone de contrôle de $P$.

**4.** Plus généralement, on considère un polygone de contrôle $\{b_0, \ldots, b_n\}$ avec $b_i = \begin{pmatrix} x_i \\ y_i \end{pmatrix}$ où $x_i = i/n$ et $y_i > 0$. Montrer que la courbe de Bézier correspondante $P(t) = \begin{cases} x(t) = t \\ y(t) \end{cases}$, $t \in [0,1]$ vérifie $y(t) > 0$. Y a-t-il une réciproque ?

**5.** Étudier les variations de $v$ et montrer que la fonction $v$ est décroissante.

**6.** Déterminer le polygone de contrôle $\{c_0, c_1, c_2, c_3\}$ de la courbe $Q_1(t)$ interprétée comme une courbe de Bézier.

**7.** Dessiner courbe et polygone de contrôle de $Q_1$.

**8.** Plus généralement, on considère un polygone de contrôle $\{c_0, \ldots, c_n\}$ avec $c_i = \begin{pmatrix} u_i \\ v_i \end{pmatrix}$ où $u_i = -1 + i/n$ et la suite $(v_i)$ est décroissante. Montrer que la courbe de Bézier correspondante $Q(t) = \begin{cases} u(t) = t - 1 \\ v(t) \end{cases}$ vérifie $v'(t) \leqslant 0$, c'est-à-dire que $v$ est décroissante. Y a-t-il une réciproque ?

**9.** Étudier la régularité du raccord entre $P$ et $Q$.

## 7.5 Détermination du polygone de contrôle

Soit $x(t) = \sum_{i=0}^{n} a_i t^i$ une fonction polynômiale de $[0,1]$ à valeurs dans $\mathbb{R}$. On souhaite déterminer les coefficients $x_i$, $i = 0, \ldots, n$ tels que $x(t) = \sum_{i=0}^{n} x_i B_i^n(t)$ ; il s'agit en fait d'un changement de base dans $\mathbb{P}_n$. Ainsi, à partir d'une courbe paramétrée $b(t) = \sum_{i=0}^{n} A_i t^i$ de $[0,1]$ dans $E = \mathbb{R}^2$, polynômiale de degré inférieur ou égal à $n$, on déterminera son polygone de contrôle $\{b_0, \ldots, b_n\}$ où $b_i = (x_i, y_i)^T$.

**1.** En calculant les dérivées successives $x^{(r)}(0)$, pour $r = 0, \ldots, n$, montrer que $x_0 = a_0$, $\Delta x_0 = a_1/n$,..., $\binom{n}{r} \Delta^r x_0 = a_r$ où $\Delta^0 x_i = x_i$ et $\Delta^r x_i = \Delta^{r-1} x_{i+1} - \Delta^{r-1} x_i$ pour $r \geqslant 1$.

On en déduit ainsi que si $x = (x_0, \ldots, x_n)^T$ et $a = (a_0, \ldots, a_n)^T$, alors

$$UVMVx = a$$

où $U$ et $V$ sont les matrices diagonales $\text{diag}(U) = \left(\binom{n}{0}, \binom{n}{1}, \ldots, \binom{n}{n}\right)$,
$\text{diag}(V) = (1, -1, \ldots, (-1)^n)$ et

$M$ la matrice triangulaire $M = \begin{pmatrix} 1 & & & & & \\ 1 & 1 & & & & \\ . & . & \ddots & & & \\ \binom{r}{0} & \binom{r}{1} & \ldots & \binom{r}{r} & & \\ . & . & . & . & \ddots & \\ \binom{n}{0} & \binom{n}{1} & \binom{n}{2} & \ldots & \binom{n}{n-1} & \binom{n}{n} \end{pmatrix}$.

**2.** Pour $n$ donné, écrire un premier programme `Matlab` qui construit les matrices $V$, $M$ et $U$. Sauvegarde sous `polyg1.m`. Test avec $n = 4$.

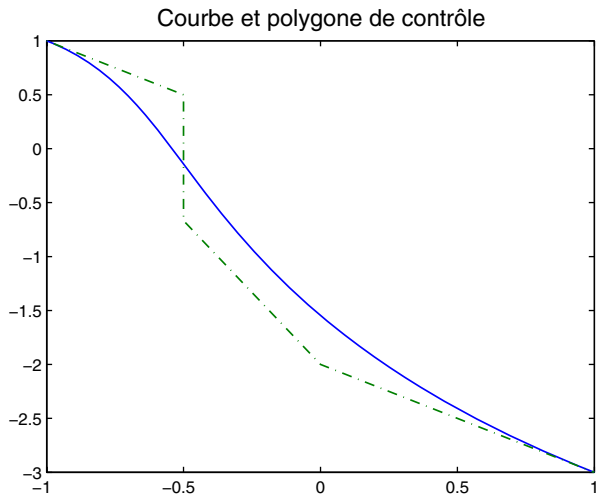
```
>> polygn1
M =
     1     0     0     0     0
     1     1     0     0     0
     1     2     1     0     0
     1     3     3     1     0
     1     4     6     4     1

V =
     1     0     0     0     0
     0    -1     0     0     0
     0     0     1     0     0
     0     0     0    -1     0
     0     0     0     0     1

U =
     1     0     0     0     0
     0     4     0     0     0
     0     0     6     0     0
     0     0     0     4     0
     0     0     0     0     1
```

**3.** Compléter pour que, étant donnée une suite finie $(a_i)$ de réels, le programme détermine $n$, puis calcule les coefficients $x_i$. Sauvegarde sous `polyg2.m`. Test avec $a = [-1, 2, -3, 4, -1]$.

```
>> polyg2
x =
   -1.0000
   -0.5000
   -0.5000
         0
    1.0000
```

**4.** Étant donnée une suite finie $(A_i)$ dans $\mathbb{R}^2$, écrire un programme qui détermine $n$, calcule les coefficients $x_i$ et $y_i$, trace la courbe $b(t) = \sum_{i=0}^{n} A_i t^i$ et le polygone de contrôle $\{b_i = (x_i, y_i)\}$. Sauvegarde sous `polyg3.m`. Test avec $A = [-1, 2, -3, 4, -1; 1, -2, -4, 2, 0]$. On pourra utiliser `polyval.m` en respectant l'ordre des coefficients.

```
>> polyg3
bx =
   -1.0000
   -0.5000
   -0.5000
         0
    1.0000

by =
    1.0000
    0.5000
   -0.6667
   -2.0000
   -3.0000
```




## DU MAL À DÉMARRER ?

### 7.1 Algorithme de de Casteljau

**1.** Une récurrence sur le calcul de $b_0^r(0)$ et $b_{n-r}^r(1)$.

**2.** Une fonction est affine si et seulement si elle conserve le barycentre.

### 7.2 Polynômes de Bernstein

**1.** Montrer que $\{B_i^n\}_{i=0,\dots,n}$ est un système libre de $\mathbb{P}_n$.

**2.** (c) Développer $(x+y)^n$, puis dériver par rapport à $x$ pour la seconde formule.

### 7.3 Raccord entre les courbes

**1.** Utiliser (7.3).

### 7.4 Contrainte de formes

**4.** Utiliser (7.2).

**8.** Utiliser (7.3).

### 7.5 Détermination du polygone de contrôle

**1.** Utiliser (7.3) et (7.4).

## CORRIGÉS DES EXERCICES

### 7.1 Algorithme de de Casteljau

**1.** Par récurrence, à chaque étape $r$ de 0 à $n$, $b_0^r(0) = b_0$ de sorte que $b^n(0) = b_0$ et de même $b^n(1) = b_n$. C'est dire que la courbe part du point $b_0$ et arrive en $b_1$.

**2.** Par définition une transformation affine $\varphi$ de $E$ conserve le barycentre si bien qu'à chaque étape $r$ et pour tout $t$ de $[0,1]$ et tout $i$ de 0 à $n-r$ :

$$\varphi(b_i^r(t)) = (1-t)\varphi(b_i^{r-1}(t)) + t\varphi(b_{i+1}^{r-1}(t)).$$

Donc par récurrence, $\varphi(b^n(t)) = \mathcal{B}(b_0,\dots,n;t)$.

➤ Programmation

```
function C=c_bezier(pc,m)
% m points de la courbe a determiner, N=n+1 points de controle
t=0:1/(m-1):1;
N=length(pc(1,:));
dim=length(pc(:,1));
tt=ones(dim,1)*t;
for i =1:N
    b(1:dim,1:m,i)=pc(:,i)*ones (1,m);
end;
for r=1:N-1
    bb=b;
    for i=1:N-r
```

```
    b(:,:,i)=(1-tt).*bb(:,:,i)+tt.*bb(:,:,i+1);
    end
end
C=b(:,:,1);
```

`trace.m`

```
%pc=[0,1,2,3,4,8;0,1.21,-1.1,0.9,-1.3,0.8];
pc=[0,1,2,3,4,8;0,1,-1,0,-1,1;1,2,-3,4,5,-3];
m=500
C=c_bezier(pc,m);
dim = length(pc(:,1));
if dim==2
    plot(C(1,:),C(2,:),pc(1,:),pc(2,:),'.-')
else
    plot3(C(1,:),C(2,:),C(3,:),pc(1,:),pc(2,:),pc(3,:),'.-')
    zlabel('z','Fontsize',14)
end
xlabel('x','Fontsize',14)
ylabel('y','Fontsize',14)
title('Courbe de Bezier et polygone de controle','Fontsize',14)
```

### Ce qu'il faut retenir de cet exercice

On aura noté l'utilisation du `if, else` qui permet grâce à `dim = length(pc( :,1))` de choisir le tracé suivant la dimension.

## 7.2 Polynômes de Bernstein

**1.** Notons que $B_i^n(0) = \delta_{i0}$ i.e. $B_i^n(0) = O$ si $i \neq 0$ et $B_0^n(0) = 1$. La dimension de $\mathbb{P}_n$ étant $n+1$, pour montrer que $\{B_i^n\}_{i=0,\ldots,n}$ forme une base, il suffit de montrer que le système est libre. Supposons que $\sum_{i=0}^{n} \lambda_i B_i^n(t) = 0$, alors en $t = 0$, il vient $\lambda_0 = 0$. Nous avons maintenant $\sum_{i=1}^{n} \lambda_i B_i^n(t) = 0$. Pour $i \geqslant 1$, $t$ est une racine d'ordre $i$ du polynôme $B_i^n$. En simplifiant la somme précédente par $t$ puis en prenant à nouveau $t = 0$, nous obtenons $\lambda_1 = 0$. De même, par récurrence tous les $\lambda_i$ sont nuls.

Rappelons que pour tout $n \in \mathbb{N}$ et tout $i \notin \{0, \ldots, n\}$ alors $B_i^n(t) = 0$. Soit $n \geqslant 1$.

Pour $i \notin \{0, \ldots, n\}$, $B_i^n(t) = 0 = (1-t)B_i^{n-1}(t) + t\,B_{i-1}^{n-1}(t)$.

$B_0^n(t) = (1-t)^n = (1-t)B_0^{n-1}(t) = (1-t)B_0^{n-1}(t) + t\,B_{-1}^{n-1}(t)$ puisque $B_{-1}^{n-1}(t) = 0$ et de même $B_n^n(t) = (1-t)B_n^{n-1}(t) + t\,B_{n-1}^{n-1}(t)$.

Si $i \in \{1, \ldots, n-1\}$,

$$\begin{aligned}(1-t)B_i^{n-1}(t) + tB_{i-1}^{n-1}(t) &= (1-t)\binom{n-1}{i}t^i(1-t)^{n-1-i} + t\binom{n-1}{i-1}t^{i-1}(1-t)^{n-1-i+1} \\ &= \left(\binom{n-1}{i} + \binom{n-1}{i-1}\right)t^i(1-t)^{n-i} \\ &= \binom{n}{i}t^i(1-t)^{n-i} = B_i^n(t).\end{aligned}$$

**2.** (a) et (b) sont élémentaires et $\sum_{i=0}^{n} B_i^n(t) = [t + (1-t)]^n = 1$ (Formule de Newton).

Sachant que $(x+y)^n = \sum_{i=0}^{n}\binom{n}{i}x^i y^{n-i}$, en dérivant par rapport à $x$, puis en multipliant par $x$, il vient : $nx(x+y)^{n-1} = \sum_{i=1}^{n} i\binom{n}{i}x^i y^{n-i}$. On prend ensuite $x = t$, $y = 1-t$, on divise par $n$ et on commence la somme à $i = 0$. On obtient $t = \sum_{i=0}^{n} i\frac{i}{n}B_i^n(t)$. On a ainsi écrit le polynôme $t$ dans la base de Bernstein.

**3.** Soit $\{b_0, \ldots, b_n\}$ un polygone de $E$. Nous allons montrer par récurrence finie sur $r = 0, \ldots, n$ que pour tout $t \in [0,1]$, $b_i^r(t) = \sum_{j=0}^{r} b_{i+j}B_j^r(t)$ pour $i = 0, \ldots, n-r$.

Pour $r = 0$, par définition, $b_i^0(t) = b_i = \sum_{j=0}^{0} b_{i+j}B_j^0(t)$.

Supposons que pour $r \geqslant 1$, $b_i^{r-1}(t) = \sum_{j=0}^{r-1} b_{i+j}B_j^{r-1}(t)$ pour $i = 0, \ldots, n-r+1$, rappelons que $b_i^r(t) = (1-t)b_i^{r-1}(t) + tb_{i+1}^{r-1}(t)$, pour $i = 0, \ldots, n-r$, si bien que

$$\begin{aligned}b_i^r(t) &= (1-t)\sum_{j=0}^{r-1} b_{i+j}B_j^{r-1}(t) + t\sum_{j=0}^{r-1} b_{i+1+j}B_j^{r-1}(t) \\ &= b_i(1-t)B_0^{r-1}(t) + \sum_{k=1}^{r-1} b_{i+k}\left((1-t)B_k^{r-1}(t) + tB_{k-1}^{r-1}(t)\right) + b_{i+r}tB_{r-1}^{r-1}(t) \\ &= b_iB_0^r(t) + \sum_{k=1}^{r-1} b_{i+k}B_k^t(t) + b_{i+r}B_r^r(t) = \sum_{k=0}^{r} b_{i+k}B_k^t(t)\end{aligned}$$

À la ligne 2, on a posé successivement $k = j$ puis $k = j + 1$.
La démonstration est ainsi terminée. On a alors la courbe de Bézier sous la forme explicite

$$b^n(t) = b_0^n(t) = \sum_{j=0}^{n} b_j B_j^n(t).$$

**4.** Pour $n \geqslant 1$ et $i \in \{1, \ldots, n-1\}$,

$$\begin{aligned}
\frac{dB_i^n}{dt}(t) &= i\binom{n}{i} t^{i-1}(1-t)^{n-i} - (n-i)\binom{n}{i} t^i (1-t)^{n-i-1} \\
&= n\frac{(n-1)!}{(i-1)!(n-1-i+1)!} t^{i-1}(1-t)^{n-1-i+1} - n\frac{(n-1)!}{i!(n-1-i)!} t^{i-1}(1-t)^{n-1-i} \\
&= n(B_{i-1}^{n-1}(t) - B_i^{n-1}(t))
\end{aligned}$$

Pour $i = 0$ ou $i = n$, la démonstration se simplifie car $B_{-1}^{n-1}(t) = 0$ puis $B_n^{n-1}(t) = 0$ et pour $i \notin \{0, \ldots, n\}$, tous les termes sont nuls.
On peut donc conclure que pour $n \geqslant 1$ et $i \in \mathbb{Z}$,

$$\frac{dB_i^n}{dt}(t) = n(B_{i-1}^{n-1}(t) - B_i^{n-1}(t))$$

Puisque $B_{-1}^{n-1} = B_n^{n-1} = 0$, on en déduit que

$$\begin{aligned}
\frac{db^n}{dt}(t) &= \sum_{j=0}^{n} b_j \frac{B_j^n}{dt}(t) = n\sum_{j=0}^{n} b_j (B_{j-1}^{n-1}(t) - B_j^{n-1}(t)) \\
&= n\left(\sum_{k=0}^{n-1} b_{k+1}(B_k^{n-1}(t) - \sum_{k=0}^{n-1} b_k (B_k^{n-1}(t)\right) = n\sum_{k=0}^{n-1} \Delta b_k B_k^{n-1}(t)
\end{aligned}$$

avec $\Delta b_k = b_{k+1} - b_k$.
L'hodographe est une courbe de Bézier de polygone de contrôle $\{n\Delta b_0, \ldots, n\Delta b_{n-1}\}$.

Par récurrence, pour $p = 1, \ldots, n$, $\dfrac{d^p b^n}{dt^p}(t) = \dfrac{n!}{(n-p)!} \displaystyle\sum_{k=0}^{n-p} \Delta^p b_k B_k^{n-p}(t)$ où $\Delta^p b_k = \Delta^{p-1} b_{k+1} - \Delta^{p-1} b_k$.

## 7.3 Raccords entre des courbes

**1.** On considère le polygone de contrôle formé des 4 points $a_i, i = 0, \cdots, 3$ suivants :

$$a_0 = (-1, 0), a_1 = (-1, 1), a_2 = (1, 0), a_3 = (1, 1).$$

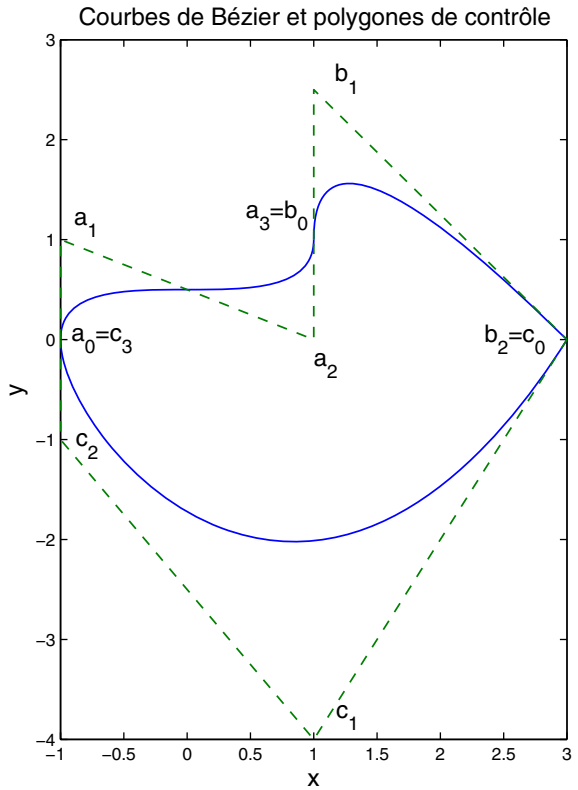
Soit $P_3(t) = \mathcal{B}(\{a_0, a_1, a_2, a_3\}, t)$ la courbe de Bézier associée. On sait que la courbe $P_3'(t)$ est associée au polygone de contrôle $3\{\Delta a_0, \Delta a_1, \Delta a_2\}$ où $\Delta a_i = a_{i+1} - a_i$, et de même pour le polygone associé à $P''_3$ qui est $6\{\Delta^2 a_0, \Delta^2 a_1\}$. Donc :

$$\{\Delta a_0, \Delta a_1, \Delta a_2\} = \{(0,1),(2,-1),(0,1)\}, \{\Delta^2 a_0, \Delta^2 a_1\} = \{(2,-2),(-2,2)\}.$$

La courbe $P''_3$ est une courbe de degré 1 donc son graphe est un segment joignant les 2 points $6(2,-2) = (12,-12)$ et $6(-2,2) = (-12,12)$.

**2.** Si on veut prolonger $P_3$ en une courbe $Q_2$ de polygone de contrôle $\{b_0 = a_3, b_1, b_2\}$, le raccord $\mathcal{C}^1$ impose : $3\Delta a_2 = 2\Delta b_0$, soit $3(a_3 - a_2) = 2(b_1 - b_0)$. En calculant indépendamment pour les abscisses et les ordonnées, on trouve : $b_1 = (1, 5/2)$.

**3.** Si on veut prolonger $P_3$ à gauche en une courbe $R_3$ de polygone de contrôle $\{c_0, c_1, c_2, c_3 = a_0\}$, le raccord $\mathcal{C}^2$ impose : $3\Delta a_0 = 3\Delta c_2$ et $6\Delta^2 a_0 = 6\Delta^2 c_1$
soit $3(a_1 - a_0) = 3(c_3 - c_2)$ et $6(a_0 - 2a_1 + a_3) = 6(c_1 - 2c_2 + c_3)$. En calculant indépendamment pour les abscisses et les ordonnées, on obtient successivement : $c_2 = (-1,-1), c_1 = (1,-4)$.



## 7.4 Contraintes de formes

**1.** Pour $t \in [0,1]$, on considère la courbe paramétrée d'équations

$$P_1(t) = \begin{cases} x(t) = t \\ y(t) = 1 - 3t + 9t^3 \end{cases} .$$

On remarque que cette courbe est le graphe d'une fonction $y = \varphi(x)$.
$y'(t) = -3 + 27t^2 = 3(3t-1)(3t+1)$, d'où le tableau de variations de la fonction $y$ :

| $t$ | 0 | | $1/3$ | | 1 |
|---|---|---|---|---|---|
| $y'(t)$ | $-$ | | 0 | | $+$ |
| $y(t)$ | 1 | $\searrow$ | $1/3$ | $\nearrow$ | 7 |

On note que sur $[0,1]$, $y(t) \geqslant 1/3 > 0$.

**2.** On rappelle que pour déterminer le polygone de contrôle $\{b_0, b_1, b_2, b_3\}$ de $P_1$ qui est de degré 3, on a les équations suivantes : $\begin{cases} P_1(0) = b_0 \\ P_1'(0) = 3(b_1 - b_0) \\ P_1(1) = b_3 \\ P_1'(1) = 3(b_3 - b_2) \end{cases}$ qui permettent de déduire :

$$b_0 = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, b_1 = \begin{pmatrix} 1/3 \\ 0 \end{pmatrix}, b_2 = \begin{pmatrix} 2/3 \\ -1 \end{pmatrix}, b_3 = \begin{pmatrix} 1 \\ 7 \end{pmatrix}$$

**4.** Plus généralement, on considère un polygone de contrôle $\{b_0, \ldots, b_n\}$ avec $b_i = \begin{pmatrix} x_i \\ y_i \end{pmatrix}$ où $x_i = i/n$ et $y_i > 0$.

La courbe de Bézier a pour équation $\begin{cases} x(t) = t \\ y(t) = \displaystyle\sum_{i=0}^{n} y_i B_i^n(t) \end{cases}$ $t \in [0,1]$. Sachant que sur $[0,1]$ les $B_i^n$ sont positives, on en déduit que $y(t) \geqslant 0$. En fait cette inégalité est stricte car les fonctions $B_i^n$ se s'annulent pas toutes au même $t$ puisque $\displaystyle\sum_{i=0}^{n} B_i^n(t) = 1$

La fonction $P_1$ permet de montrer que $y(t) > 0$ bien que $y_2 < 0$. Il n'y a pas de réciproque au résultat précédent.

**5.** Pour $t \in [0,1]$, on considère la courbe paramétrée d'équations

$$Q_1(t) = \begin{cases} u(t) = t - 1 \\ v(t) = 2 - 3t + 6t^2 - 4t^3 \end{cases}.$$

$v'(t) = -3 + 12t - 12t^2 = -3(2t-1)^2 \leqslant 0$ pour $t \in [0,1]$, d'où le tableau de variations de la fonction $v$ :

| $t$ | 0 | | 1 |
|---|---|---|---|
| $v'(t)$ | | $-$ | |
| $v(t)$ | 2 | $\searrow$ | 1 |

On note que sur $[0,1]$, $v$ est décroissante.

**6.** On applique la même méthode que précédemment pour déterminer le polygone de contrôle $\{c_0, c_1, c_2, c_3\}$ de $Q_1$ qui est de degré 3, on a les équations suivantes : $\begin{cases} Q_1(0) = c_0 \\ Q_1'(0) = 3(c_1 - c_0) \\ Q_1(1) = c_3 \\ Q_1'(1) = 3(c_3 - c_2) \end{cases}$

qui permettent de déduire :

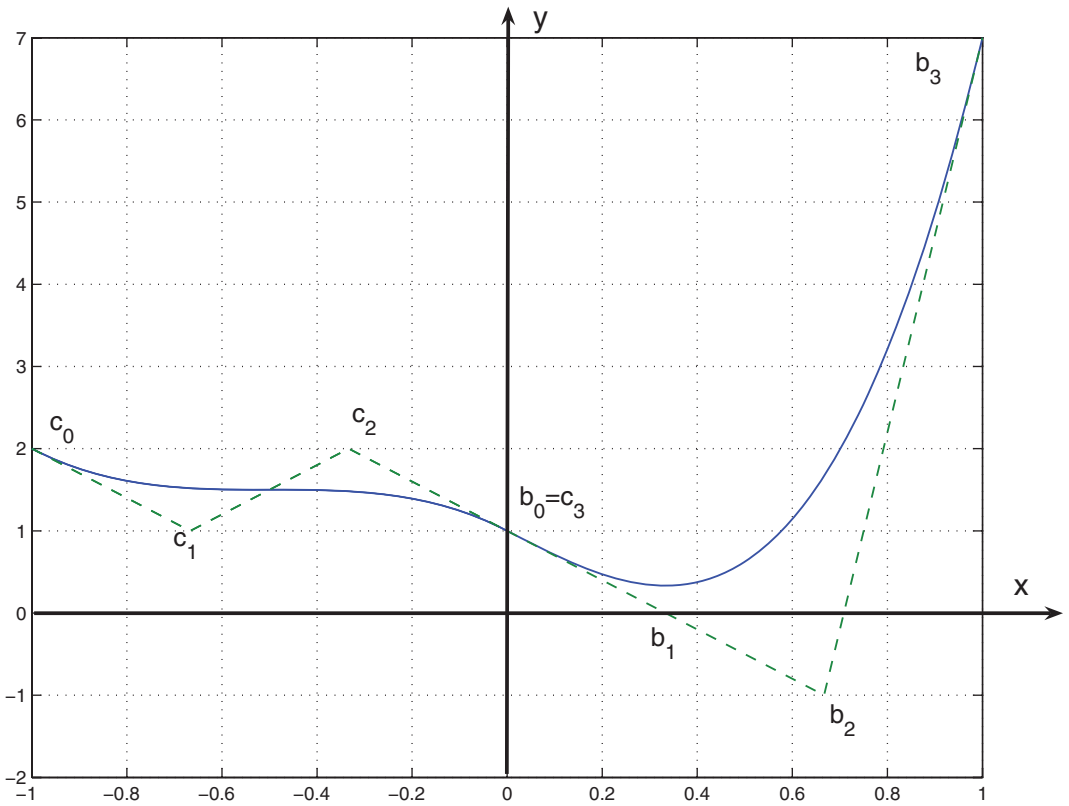
$$c_0 = \begin{pmatrix} -1 \\ 2 \end{pmatrix}, c_1 = \begin{pmatrix} -2/3 \\ 1 \end{pmatrix}, c_2 = \begin{pmatrix} -1/3 \\ 2 \end{pmatrix}, c_3 = \begin{pmatrix} 0 \\ 1 \end{pmatrix} = b_0$$

**8.** Plus généralement, on considère un polygone de contrôle $\{c_0, \ldots, c_n\}$ avec $c_i = \begin{pmatrix} u_i \\ v_i \end{pmatrix}$ où $u_i = -1 + i/n$ et la suite $v_i$ est décroissante c'est-à-dire que $v_{i+1} - v_i \leqslant 0$ pour $i = 0, \ldots, n-1$.

La courbe de Bézier a pour équation $\begin{cases} u(t) = t - 1 \\ v(t) = \displaystyle\sum_{i=0}^{n} v_i B_i^n(t) \end{cases}$ $t \in [0,1]$. On a alors $v'(t) = n \displaystyle\sum_{i=0}^{n-1} (v_{i+1} - v_i) B_i^{n-1}(t)$. Sur $[0,1]$ les $B_i^{n-1}$ sont positives. On en déduit que $v'(t) \leqslant 0$ donc $v$ est décroissante.

La fonction $Q_1$ permet de montrer que $v' \leqslant 0$ bien que $y_2 - y_1 > 0$. Il n'y a pas de réciproque au résultat précédent.

**9.** On a $c_3 = b_0$ puis $b_1 - b_0 = c_3 - c_2$ donc le raccord entre les 2 courbes est $\mathcal{C}^1$, par contre $\Delta^2 b_0 = b_2 - 2b_1 + b_0 = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$ et $\Delta^2 c_1 = c_3 - 2c_2 + c_1 = \begin{pmatrix} 0 \\ -12 \end{pmatrix}$. Ces valeurs différentes montrent que le raccord n'est pas $\mathcal{C}^2$. On pouvait aussi comparer $y''(0)$ et $v''(1)$.




## 7.5 Détermination du polygone de contrôle

**1.** En utilisant (7.4), on obtient

$$x^{(r)}(0) = \frac{d^r b^n}{dt^r}(0) = \frac{n!}{(n-r)!}\Delta^r x_0$$

où $\Delta^r x_k = \Delta^{r-1}x_{k+1} - \Delta^{r-1}x_k$ car $B_k^{n-r}(0) = 0$ si $k \neq 0$. Par ailleurs si $x(t) = \sum_{i=0}^{n} a_i t^i$, on a $x^{(r)}(0) = r!a_r$. Donc pour $r = 0, \ldots, n$, on obtient $a_r = \binom{n}{r}\Delta^r x_0$. Par récurrence, on obtient

$$\begin{aligned} x_i &= 1x_i \\ \Delta x_i &= x_{i+1} - x_i, \\ \Delta^2 x_i &= x_{i+2} - 2x_{i+1} + x_i, \\ &\vdots \\ \Delta^r x_i &= \binom{r}{r} x_{i+r} - \binom{r}{r-1} x_{i+r-1} + \ldots + (-1)^r \binom{r}{0} x_i, \end{aligned}$$

alternance de signe et coefficients binômiaux, ce qui permet d'écrire $UVMx = a$ où les matrices $U, V, M$ sont définies précédemment.

**2.** polyg1.m

```
n=4;
N=n+1;
M=eye(N);
M(:,1)=ones(N,1);
for i=2:N
        M(i,2:i)=M(i-1,1:i-1)+M(i-1,2:i);
end
M
V(1:2:N)=1;
V(2:2:N)=-1;
V=diag(V)
U=diag(M(N,:))
```

### Ce qu'il faut retenir de cet exercice

La matrice $M$ est construite en utilisant le triangle de Pascal qui donne les coefficients binômiaux ligne par ligne grace à $\binom{r+1}{i} = \binom{r}{i} + \binom{r}{i-1}$ pour $i = 1, \ldots, r$.

**4.** polyg3.m

```
A=[-1,2,-3,4,-1;1,-2,-4,2,0];
N=length(A(1,:));
n=N-1
%en fait il y a N=n+1 points de controle
t=0:1/500:1;
cx=polyval(A(1,N:-1:1),t);
cy=polyval(A(2,N:-1:1),t);
M=eye(N);
M(:,1)=ones(N,1);
for i=2:N
        M(i,2:i)=M(i-1,1:i-1)+M(i-1,2:i);
end
U=diag(M(N,:));
V(1:2:N)=1;
V(2:2:N)=-1;
M=U*diag(V)*M*diag(V)

bx=M\A(1,:)'
by=M\A(2,:)'
tt=0:1/(N-1):1;
plot(cx,cy,bx,by,'-.')
title('Courbe␣et␣polygone␣de␣controle','Fontsize',14)
```

### Ce qu'il faut retenir de cet exercice

On notera qu'il s'agit en fait d'un changement de base pour une fonction polynômiale, on passe de la base canonique à la base de Bernstein en déterminant les $x_i$ en fonction des $a_j$, chaque variable $x$ ou $y$ étant traitée de manière indépendante.

# 8

# Équations différentielles, méthodes à un pas

## RAPPEL DE COURS

Soit $I = [t_0, t_0+T]$ un intervalle fermé borné de $\mathbb{R}$. On se donne une fonction $f$ définie et continue sur $I \times \mathbb{R}^m$ à valeurs dans $\mathbb{R}^m$. Pour $\eta \in \mathbb{R}^m$, condition initiale, on cherche une fonction $y$ définie et dérivable sur $I$ à valeurs dans $\mathbb{R}^m$ telle que

$$y'(t) = f(t, y(t)), \forall t \in I \tag{8.1}$$
$$y(t_0) = \eta. \tag{8.2}$$

Il s'agit d'un système différentiel du premier ordre.

*S'il existe un réel L et une norme sur $\mathbb{R}^m$ tels que*

$$\forall t \in I, \forall y, z \in \mathbb{R}^m, \ \|f(t,y) - f(t,z)\| \leqslant L\|y - z\|, \ \textit{(Condition de Lipschitz)}$$

*alors le problème* (8.1)-(8.2) *admet une solution unique.*

**Schémas numériques :** À partir d'une subdivision $t_0 < t_1 < \ldots < t_n = t_0 + T$, on pose $h_i = t_{i+1} - t_i$, $H = \max h_i$. Pour une fonction $\phi$ continue de $I \times \mathbb{R}^m \times R$ à valeurs dans $\mathbb{R}^m$ nous construisons la suite finie $u_0 \in \mathbb{R}^m$, et pour $i = 0, \ldots, n-1$, $u_{i+1} = u_i + h_i\phi(t_i, u_i, h_i)$.

Nous construisons aussi des suites « perturbées » : $v_0 \in \mathbb{R}^m$, $v_{i+1} = v_i + h_i\phi(t_i, v_i, h_i) + \varepsilon_i$. Le schéma est *stable* s'il existe une constante $c$ indépendante des suites $(u_i)$, $(v_i)$ telle que : $\max_{i=0,\ldots,n} |u_i - v_i| \leqslant C \left( |u_0 - v_0| + \sum_{j=0}^{n} |\varepsilon_j| \right)$. *Le schéma est stable si et seulement si il existe* $\Lambda > 0$ *tel que :* $\forall t \in I, \forall u, v \in \mathbb{R}^m, \forall h \in [0, H], \ \|\phi(t,u,h) - \phi(t,v,h)\| \leqslant \Lambda\|y - z\|$.

Si $y$ est une solution de (8.1), l'*erreur de consistance* est définie par

$$\varepsilon(y) = \sum_{i=0}^{n-1} \|y(t_{i+1}) - y(t_i) - h_i\phi(t_i, y(t_i), h_i)\|.$$

La méthode est consistante si $\varepsilon(y)$ tend vers 0 quand $H$ tend vers 0. *Si la méthode est stable et consistante alors elle est convergente i.e lorsque* $u_0$ *tend vers* $\eta$ *et* $H$ *tend vers* 0*, alors* $\max \|y(t_i) - u_i\|$ *tend vers 0.*

Le schéma numérique est d'*ordre* $p > 0$ s'il existe un réel $K$ (ne dépendant que de $y$ et de $\phi$) tel que $\varepsilon(y) \leqslant K H^p$ dès que $y$ est une solution de classe $\mathcal{C}^{p+1}$ de (8.1). *Soit* $y$ *la solution de* (8.1)-(8.2). *Si le schéma numérique est stable et d'ordre* $p$*, si* $f \in \mathcal{C}^p(I \times \mathbb{R}^m, \mathbb{R}^m)$ *alors*

$$\forall i = 0, \ldots, n,\ \|y(t_i) - u_i\| \leqslant e^{\Lambda T} \left( \|\eta - u_0\| + K H^p \right).$$

Rappelons trois schémas numériques : la méthode d'Euler explicite, la méthode de Heun (ou schéma de Runge-Kutta d'ordre 2) et la méthode de Runge-Kutta classique, d'ordre 4.

**Méthode d'Euler :** $\phi(t, y, h) = f(t, y)$,

$$\begin{cases} u_0 = \eta \\ u_{i+1} = u_i + h f(t_i, u_i),\ i = 0, \ldots, n-1 \end{cases} \tag{8.3}$$

**Méthode de Heun :** $\phi(t, y, h) = \dfrac{1}{2}\left(f(t, y) + f(t + h, y + h f(t, y))\right)$.

$$\begin{cases} u_0 = \eta \\ u_{i+1} = u_i + \dfrac{h}{2}\left(f(t_i, u_i) + f(t_{i+1}, u_i + h_i f(t_i, u_i))\right),\ i = 0, \ldots, n-1 \end{cases} \tag{8.4}$$

**Méthode de Runge-Kutta :**

$$\begin{cases} u_0 = \eta \\ u_{i+1} = u_i + \dfrac{h_i}{6}\left( f(t_i, u_{i,1}) + 2 f(t_i + \dfrac{h_i}{2}, u_{i,2}) + 2 f(t_i + \dfrac{h_i}{2}, u_{i,3}) + f(t_{i+1}, u_{i,4}) \right), \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad i = 0, \ldots, n-1 \end{cases} \tag{8.5}$$

où

$$u_{i,1} = u_i,\ u_{i,2} = u_i + \frac{h_i}{2} f(t_i, u_{i,1}),\ u_{i,3} = u_i + \frac{h_i}{2} f\left(t_i + \frac{h_i}{2}, u_{i,2}\right),\ u_{i,4} = u_i + h_i f\left(t_i + \frac{h_i}{2}, u_{i,3}\right).$$

Les instructions `Matlab` habituelles pour les équations différentielles ou les systèmes du premier ordre sont `ode23, ode45, ode 113...`. Elles sont construites sur des méthodes à pas variables.

Plutôt que de reprendre une partie des démonstrations, nous avons commencé directement par un exercice mettant en évidence l'importance du choix du pas. Par contre, au chapitre suivant, on retrouvera les outils utilisés : stabilité, erreur de consistance, ordre... avec des démonstrations.

Le deuxième exercice montre comment se ramener d'une équation du second ordre à un système du premier ordre puis donne la solution exacte (mais formelle) d'une équation linéaire de ce type avec conditions initiales. Les deux exercices suivants utilisent `Matlab` pour une détermination numérique de l'erreur pour deux schémas. Enfin dans la méthode de tir, nous retrouvons une équation du second ordre avec deux conditions aux bords que nous étudierons aussi dans le chapitre 10.

## ÉNONCÉS DES EXERCICES

### 8.1 Conditions initiales et pas

Ce premier exemple est extrait du livre de Crouzeix et Mignot [6].

On considère le problème

$$\begin{cases} y'(t) = -150y(t) + 49 - 150t, \ t \in [0,1] \\ y(0) = 1/3 + \varepsilon \end{cases}$$

**1.** Déterminer la solution exacte du problème $y^\varepsilon$.

**2.** $\varepsilon$ désigne l'erreur faite sur la condition initiale. Montrer que $\|y^0 - y^\varepsilon\|_\infty \leqslant \varepsilon$. *Il n'y a donc pas d'amplification de l'erreur initiale.*

**3.** Soit $h > 0$. Pour $t$ et $t + h$ dans $[0,1]$, montrer que
$y^0(t+h) = y^0(t) + h(-150y^0(t) + 49 - 150t)$.

**4.** Soient $n \in \mathbb{N}$ avec $n \neq 0$, $h = \dfrac{1}{n}$ et $t_i = ih, \ i = 0, \ldots, n$. On détermine une solution approchée $u_i^\varepsilon$ de $y^\varepsilon(t_i)$ par la méthode d'Euler (8.3). Montrer que
$u_{i+1}^\varepsilon - y^0(t_{i+1}) = (1 - 150h)(u_i^\varepsilon - y^0(t_i))$ puis que $u_i^\varepsilon - y^0(t_i) = (1 - 150h)^i \varepsilon$ pour $i = 0, \ldots, n$.

**5.** On prend $n = 50$ et $\varepsilon = 0.01$. Calculer l'erreur $u_n^\varepsilon - y^0(1)$.

**6.** Donner une condition sur $n$ pour que $\max\limits_{i=0,\ldots,n} |u_i^\varepsilon - y^0(t_i)| \leqslant \varepsilon$.

**7.** **Programmation.** Dans `Matlab`, créer un fichier `f1.m` contenant la fonction définie par $f1(t,y) = -150y + 49 - 150t$.

**8.** Programmer la méthode de Heun (8.4) puis évaluer l'erreur $\max\limits_{i=0,\ldots,n} |u_i^\varepsilon - y^0(t_i)|$. Sauvegarde dans `RK2.m`. Test avec $n = 60$ puis $n = 100$ et $\varepsilon = 0.01$.

```
>> RK2
n =
    60

errmax =
  4.4792e+010
```

```
>> RK2
n =
   100

errmax =
    0.0100
```

*On constate le même phénomène que pour la méthode d'Euler.*

**9.** Appeler une méthode de résolution approchée de `Matlab` avec une condition initiale $u_0 = 1/3 + 0.01$. Déterminer l'erreur $\max_{i=0,\dots,n} |u_i^\varepsilon - y^0(t_i)|$ ainsi que le nombre de composantes du vecteur $t$. On pourra aussi examiner les variations du pas de temps $t_{i+1} - t_i$. Sauvegarde dans `resol1.m`.

## 8.2 Équation linéaire du second ordre

$a$ et $b$ étant 2 réels, $f$ une fonction continue sur un intervalle contenant $x_0$, on considère le problème

$$(\mathcal{P}) \ : \ \begin{cases} y''(x) + ay'(x) + by(x) = f(x), \\ y(x_0) = \alpha, \ y'(x_0) = \beta \end{cases}$$

**1.** En posant $Y(x) = \begin{pmatrix} y(x) \\ y'(x) \end{pmatrix}$, montrer que $(\mathcal{P})$ peut s'écrire

$$(\mathcal{P}') \ : \ \begin{cases} Y'(x) = AY(x) + F(x) \\ Y(x_0) = \eta \end{cases}$$

où $A$ est une matrice $2 \times 2$, $\eta = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$ et $F$ une fonction à valeurs dans $\mathbb{R}^2$.

**2.** Pour $M \in \mathbb{R}^{n\times n}$, on définit la matrice $e^{xM} = \displaystyle\sum_{k=0}^{+\infty} \frac{x^k M^k}{k!}$, série normalement convergente sur un intervalle borné.

(a) Que se passe-t-il si $M$ est diagonalisable ($M = PDP^{-1}$ avec $D$ diagonale) ?

(b) Dans le cas général, montrer que

$$\frac{de^{xM}}{dx} = Me^{xM} = e^{xM}M,$$

$$e^{(x+x')M} = e^{xM} \times e^{x'M},$$

$$(e^{xM})^{-1} = e^{-xM}$$

**3.** On revient au problème $(\mathcal{P}')$. Montrer que les solutions de $Y'(x) = AY(x)$ sont de la forme $Y(x) = e^{xA}\lambda$ où $\lambda \in \mathbb{R}^2$.

**4.** En cherchant une solution particulière sous la forme $Y(x) = e^{xA}\lambda(x)$, montrer que la solution unique de $(\mathcal{P}')$ est donnée par

$$Y(x) = e^{(x-x_0)A}\eta + \int_{x_0}^{x} e^{(x-t)A}F(t)dt.$$

L'unique solution de $(\mathcal{P})$ est alors la première composante de cette fonction vectorielle.

## 8.3 Erreur dans la méthode de Runge-Kutta

On considère le problème

$$\begin{cases} y''(t) = -y(t),\ t \in [0,\pi] \\ y(0) = 0,\ y'(0) = 1 \end{cases}$$

dont la solution est donnée par $y(t) = \sin t$.

**1.** Transformer ce problème en un système d'ordre 1 en posant $Y(t) = \begin{pmatrix} y(t) \\ y'(t) \end{pmatrix}$.

$$(\mathcal{P})\ \begin{cases} Y'(t) = F(t, Y(t)),\ t \in [0,\pi] \\ Y(0) = \eta \end{cases}$$

**2.** **Programmation.** Écrire un fichier `f2.m` qui calcule la fonction $F$.
Test `f2(1,[1 2])`.

```
>> f2(1,[1 2])
ans =
     2
    -1
```

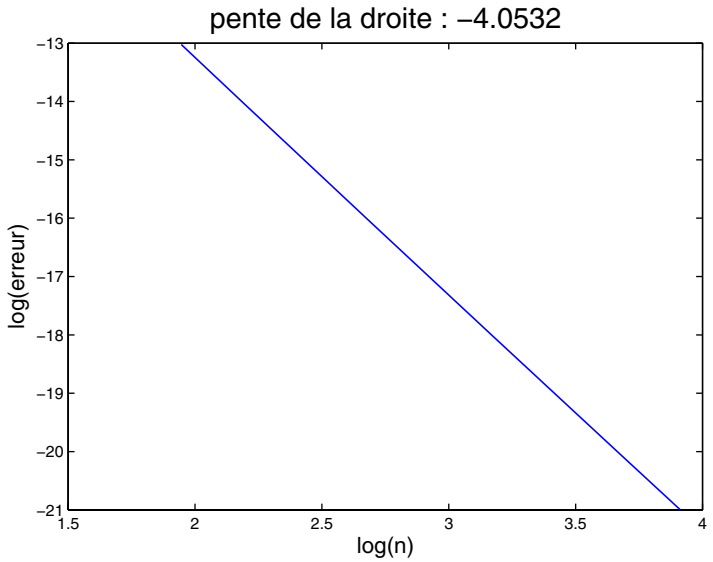
**3.** Programmer la méthode de Runge-Kutta (8.5). On créera une fonction avec en paramètres d'entrée `fichier` le nom du fichier contenant $F$, `n` le nombre de subdivisions, `eta` la condition initiale et `interv` un vecteur à 2 composantes, les 2 bornes de l'intervalle d'intégration et en sortie le tableau des abscisses $t$ et les solutions $U$. On pourra utiliser la fonction `feval.m` de `Matlab`.
Test `[t,U]=RK4('f2',5,eta,[0 1])`

**4.** Pour un $n$ donné, calculer $err = \max_{i=0,\ldots,n} |U_i^1 - sin(t_i)|$ où $U_i^1$ désigne la première composante de $U_i$. Sauvegarde dans `erreur1.m`. Test avec $n = 3$.

```
>> erreur1
n =
     3

err =
  7.7113e-005
```

**5.** Étudier l'erreur quand $n$ varie. On partira du tableau de valeurs $tn$ et on construira un tableau $taberr$ puis on tracera $\log(taberr)$ en fonction de $\log(tn)$. Sauvegarde sous `erreur2.m`. Test avec $tn = [2, 3, 5, 7, 10, 15, 20, 35, 50]$. À l'aide de l'instruction `polyfit`, on calculera la pente de la droite de régression pour conclure à l'ordre de la méthode.

```
>> erreur2
tn =
     7    10    15    20    35    50

taberr =
  1.0e-005 *
    0.2208    0.0513    0.0098    0.0031    0.0003    0.0001
```




## 8.4 Système différentiel

On considère le système différentiel

$$(\mathcal{S}) \begin{cases} y'(t) = Ay(t),\ t \in [0,1] \\ y(0) = \eta \end{cases} \quad \text{où } A = \begin{pmatrix} -1 & 1 \\ 0 & -1 \end{pmatrix},\ \eta = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$$

**1.** On rappelle que $e^{At} = \sum_{k=0}^{+\infty} \frac{(At)^k}{k!}$, série normalement convergente pour $t \in [a,b]$. Montrer que $\frac{de^{At}}{dt} = Ae^{At}$. En déduire que l'unique solution de $(\mathcal{S})$ est $yex(t) = e^{At}\eta$.

**2.** Calculer $A^k$ et montrer que la solution exacte vérifie $yex(t) = \begin{pmatrix} y_1(t) \\ y_2(t) \end{pmatrix} = \begin{pmatrix} (\alpha + \beta t)e^{-t} \\ \beta e^{-t} \end{pmatrix}$.

On souhaite approcher la solution de $(\mathcal{S})$ par une méthode d'Euler modifiée. Avec les notations habituelles, $h = 1/n$ et $t_j = jh$ pour $j = 0$ à $n$.

$$\begin{cases} U_0 = \eta \\ U_{j+1} = U_j + h\, f\left(t_j + \frac{h}{2},\ U_j + \frac{h}{2} f(t_j, U_j)\right),\ j = 0, \ldots, n-1 \end{cases}$$

qui dans le cas particulier donne comme approximation de $yex(t_j)$ le vecteur à 2 composantes $U_j$

$$U_{j+1} = \left( Id + hA + \frac{h^2}{2} A^2 \right) U_j \text{ et } U_0 = \eta$$

Évidemment avec `Matlab` les indices commenceront à 1 et iront jusqu'à $n+1$.

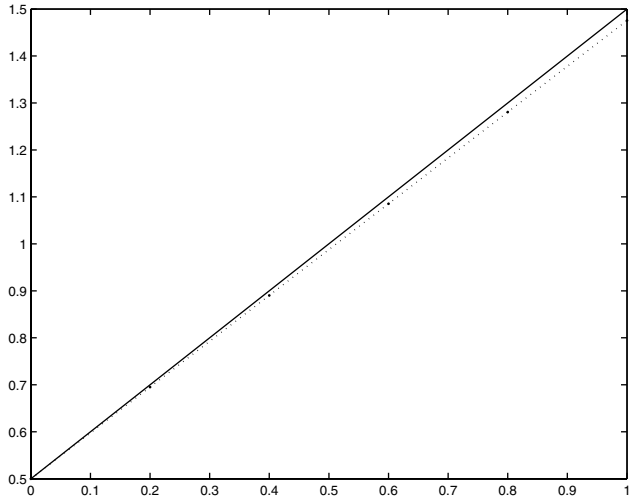
**3.** Écrire un premier programme qui construit $A$ et calcule $B = \left( Id + hA + \frac{h^2}{2} A^2 \right)$. Sauvegarder sous `SD1.m`. Test $n = 5$, afficher $B$.

```
>> SD1
n =
     5

B =
    0.8200    0.1600
         0    0.8200
```

**4.** Construire la suite des valeurs approchées $U(i,j)_{i=1,2,\ j=1,\ldots,n+1}$ avec l'initialisation $U(:,1) = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$. Sauvegarder sous `SD2.m`. Test $n = 5$, $\alpha = 1$, $\beta = 2$. Afficher `U( :,6)`.

```
>> SD2
ans =
    1.0941
    0.7415
```

**5.** Ajouter un graphe qui dessine $U(1,:) \cdot / U(2,:)$ et $yex(1,:) \cdot / yex(2,:)$ en fonction de $t$. Le tableau $yex(i,j)$ est construit à l'aide d'un programme `solex.m` appelé par `yex=solex(eta,t)`. Sauvegarder sous `SD3.m`. Test avec $n = 5$, $\alpha = 1$, $\beta = 2$.



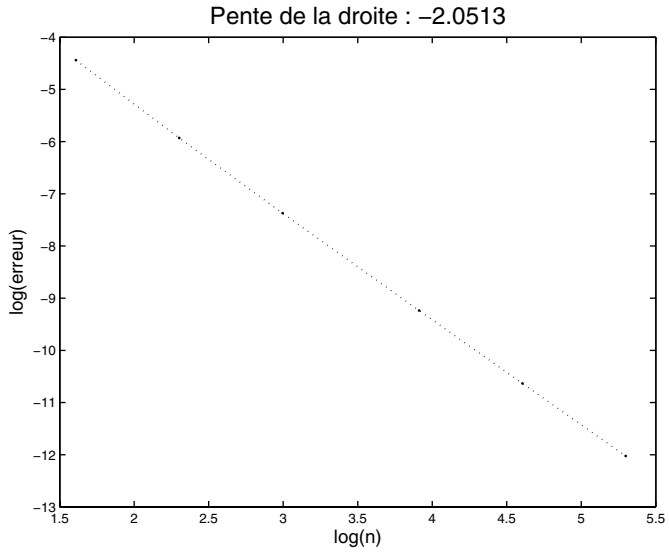
## 6. Étude d'erreur

Pour $n$ fixé, évaluer $\|U(:,j) - yex(:,j)\|_2$ pour $j = 1$ à $n+1$, puis
$err = \max_{j=1,\ldots,n+1} \|U(:,j) - yex(:,j)\|_2$. Sauvegarder sous `SD4.m`. Afficher $err$ pour $n = 10$.

```
>> SD4
ans =
    0.0027
```

**7.** Faire varier $n$ sur plusieurs valeurs entre 5 et 200 et dessiner $\log(err)$ en fonction de $\log(n)$. En déduire l'ordre de la méthode. Sauvegarder sous `SD5.m`.

```
>> SD5
tn =
     5    10    20    50   100   200

taberr =
    0.0118    0.0027    0.0006    0.0001    0.0000    0.0000
```




**8.** On considère l'équation différentielle du second ordre

$$y''(t) = -ty'(t) - y(t)\ t \in [0,1] \text{ avec } y(0) = 1,\ y'(0) = 0.$$

La solution exacte est $y(t) = e^{-t^2/2}$. Transformer cette équation en un système

$$\begin{cases} Y'(t) = A(t)Y(t) \\ Y(0) = \eta \end{cases} \quad \text{où } Y(t) = \begin{pmatrix} y(t) \\ y'(t) \end{pmatrix}.$$

Montrer que la solution approchée $U$ par la méthode d'Euler modifiée est donnée par

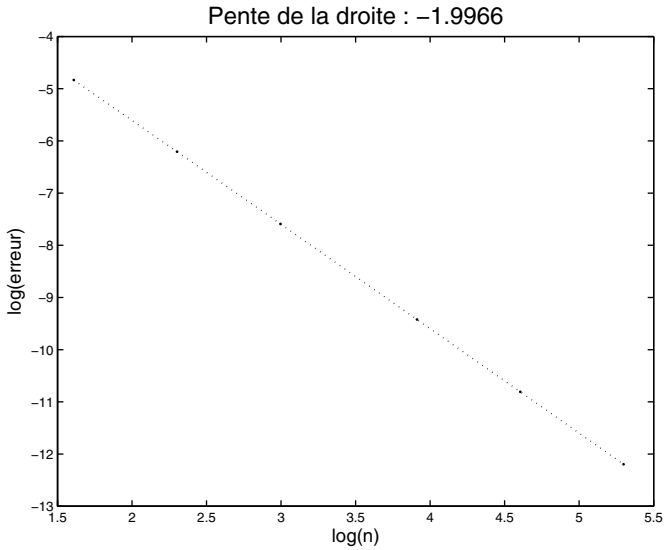
$$U_{j+1} = \left( Id + h\, A\left(t_j + \frac{h}{2}\right) + \frac{h^2}{2} A\left(t_j + \frac{h}{2}\right) A\left(t_j\right) \right) U_j \text{ et } U_0 = \eta$$

puis la construire. Sauvegarder sous `SD6.m`. Test $n = 5$ afficher $U(:, 6)$.

```
>> SD6
ans =
    0.5986
   -0.6056
```

**9.** Étudier l'erreur $\max_{j=1,\ldots,n+1} |y(t_j) - U(1, j)|$ en fonction de $n$. Sauvegarder sous `SD7.m`.

```
>> SD7
tn =
     5    10    20    50   100   200

taberr =
    0.0080    0.0020    0.0005    0.0001    0.0000    0.0000
```




## 8.5 Méthode de tir

La méthode de tir s'applique aux équations différentielles du second ordre avec conditions aux deux extrémités.

Soit le problème

$$(\mathcal{PT}) \begin{cases} y''(t) = f(t, y(t), y'(t)),\ t \in [0,1] \\ y(0) = 1,\ y(1) = a \end{cases} \tag{8.6}$$

La méthode de tir suppose que l'on sait résoudre un autre problème, de Cauchy,

$$(\mathcal{PC}) \begin{cases} y''(t) = f(t, y(t), y'(t)) \\ y(0) = 1,\ y'(0) = v. \end{cases} \tag{8.7}$$

Pour tout $v$, la solution du problème de Cauchy (8.7) donne une valeur $y_v(1) = b$. On a résolu le problème (8.6) lorsque $b = a$. La méthode de tir est un algorithme pour déterminer $v$ tel que $y_v(1) = a$.

➤ Cas de l'équation linéaire du second ordre

On note $y_0$, $y_1$, $y_2$ les solutions respectives des problèmes

$$(\mathcal{P}_0) \begin{cases} y''(t) = 1 - 2t\cos t - y(t) + ty'(t), \\ y(0) = 1,\ y(1) = \alpha. \end{cases}$$

$$(\mathcal{P}_1) \begin{cases} y''(t) = 1 - 2t\cos t - y(t) + ty'(t), \\ y(0) = 1,\ y'(0) = 0. \end{cases}$$

$$(\mathcal{P}_2) \begin{cases} y''(t) = 1 - 2t\cos t - y(t) + ty'(t), \\ y(0) = 1,\ y'(0) = 1. \end{cases}$$

Montrer qu'il existe un $\lambda \in \mathbb{R}$ tel que

$$y_0 = \lambda y_1 + (1 - \lambda) y_2.$$

➤ Programmation

On approche $y_i$ par $u_i$ pour $i = 0, 1$ ou 2 et $u_0 = \bar{\lambda} u_1 + (1 - \bar{\lambda}) u_2$ où $\bar{\lambda}$ est choisi pour que $u_0(1) = \alpha$ .

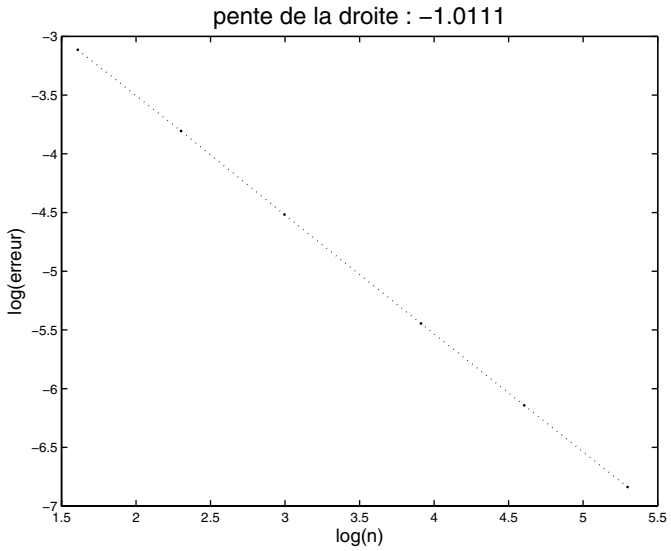
**1.** Transformer les problèmes $(\mathcal{P}_i)$ en système du premier ordre et créer un fichier `f.m` où $Y' = f(t, Y)$.

**2.** Pour $n \in \mathbb{N}$ donné, on définit $h = \dfrac{1}{n}$ et on subdivise régulièrement l'intervalle $[0, 1]$ aux points $t_j = jh$. Programmer la méthode de tir, sous forme d'un programme prenant $n$ et $\alpha$ comme argument, et fournissant $u_0(t_j)$, $0 \leqslant j \leqslant n$. Pour cela, on pourra

- Calculer $u_1(t_j)$, $1 \leqslant j \leqslant n$ et $u_2(t_j)$, $1 \leqslant j \leqslant n$ au moyen de la méthode d'Euler, en utilisant un sous programme `eul.m`, appel `[t,U]=eul(fichier,n,eta,interv)`, où `fichier` contient la fonction $f$, `n` est le nombre de subdivisions, `eta` la condition initiale et `interv` un vecteur à 2 composantes, les 2 bornes de l'intervalle d'intégration.
- Calculer $\bar{\lambda}$ en utilisant la remarque précédente et regrouper les résultats pour obtenir $u_0$ .

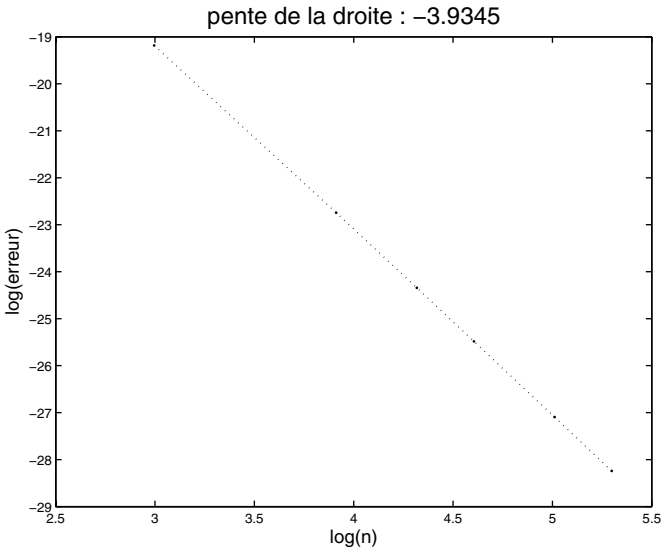
**3.** Comparer les résultats obtenus à la solution exacte. Pour $\alpha = 2(1 + \sin 1)$, la solution exacte est donnée par $t + 2\sin t + 1$. On calculera $\|y_0 - u_0\|_\infty = \max_j \|y_0(t_j) - u_0(t_j)\|$, puis on étudiera l'erreur en fonction de $n$. Sauvegarde dans `tir2.m`.

```
>> tir2
tn =
     5    10    20    50   100   200

taberr =
    0.0444    0.0222    0.0109    0.0043    0.0021    0.0011
```




**4.** Remplacer la méthode d'Euler par la méthode de Runge-Kutta en utilisant le programme `RK4.m`.

```
>> tir3
tn =
    20    50    75   100   150   200

taberr =
  1.0e-008 *
    0.4661    0.0133    0.0027    0.0009    0.0002    0.0001
```




## DU MAL À DÉMARRER ?

### 8.1 Conditions initiales et pas

**1.** Équation différentielle du premier ordre, linéaire avec second membre.

**3.** On peut faire un calcul direct ou remarquer que $\dfrac{y^0(t+h)-y^0(t)}{h} = y^{0\prime}(t)$, puisque $y^0$ est affine.

### 8.2 Équation linéaire du second ordre

**2.** (a) $M = PDP^{-1}$ alors $M^n = PD^nP^{-1}$.

**3.** Poser $Z(x) = e^{-xA}Y(x)$.

### 8.3 Erreur dans la méthode de Runge-Kutta

**1.** Si $Y(t) = \begin{pmatrix} y(t) \\ y'(t) \end{pmatrix} = \begin{pmatrix} y_1(t) \\ y_2(t) \end{pmatrix}$, calculer $Y'(t)$ en fonction de $t$, $y_1(t)$, $y_2(t)$.

### 8.4 Système différentiel

**1.** Poser $Z(t) = e^{-tA}Y(t)$.

### 8.5 Méthode de tir

**1.** $Y(t) = \begin{pmatrix} y(t) \\ y'(t) \end{pmatrix} = \begin{pmatrix} y_1(t) \\ y_2(t) \end{pmatrix}$, calculer $Y'(t)$.

## CORRIGÉS DES EXERCICES

### 8.1 Conditions initiales et pas

Pour résoudre

$$y'(t) = -150y(t) + 49 - 150t, \tag{8.8}$$

on résout tout d'abord l'équation homogène $y'(t) + 150y(t) = 0$ dont les solutions forment un espace vectoriel de $\mathcal{C}^1(\mathbb{R})$ et vérifient $y(t) = \lambda e^{-150t}$. En cherchant une solution particulière sous forme d'un polynôme de degré 1, on trouve $y(t) = 1/3 - t$. Les solutions de (8.8) sont donc définies sur $\mathbb{R}$ et vérifient $y(t) = 1/3 - t + \lambda e^{-150t}$. Si on précise la condition initiale $y(0) = \dfrac{1}{3} + \varepsilon$, on trouve une solution unique définie par $y^\varepsilon(t) = 1/3 - t + \varepsilon e^{-150t}$.
On a alors $y^\varepsilon(t) - y^0(t) = +\varepsilon e^{-150t}$, si bien que pour $t \in [0,1]$, $|y^\varepsilon(t) - y^0(t)| \leqslant \varepsilon$, donc $\|y^\varepsilon - y^0\|_\infty = \max\limits_{t\in[0,1]} |y^\varepsilon(t) - y^0(t)| \leqslant \varepsilon$.

Comme $y^0$ est une fonction affine, il vient $y^0(t+h) - y^0(t) = hy^{0\,\prime}(t)$. Puisque $y^0$ est solution de (8.8), on obtient

$$y^0(t+h) = y^0(t) + h(-150y^0(t) + 49 - 150t). \tag{8.9}$$

La méthode d'Euler (8.3) permet de déterminer une solution approchée $u_i^\varepsilon$ de $y^\varepsilon(t_i)$ par

$$\begin{cases} u_0^\varepsilon = \dfrac{1}{3} + \varepsilon \\ u_{i+1}^\varepsilon = u_i^\varepsilon + h(-150u_i^\varepsilon + 49 - 150t_i),\ i = 0, \ldots, n-1 \end{cases}$$

En retranchant (8.9) à la dernière équation pour $t = t_i$, on obtient

$$u_{i+1}^\varepsilon - y^0(t_{i+1}) = (1 - 150h)(u_i^\varepsilon - y^0(t_i)),\ i = 0, \ldots, n-1;$$

il s'agit d'une suite géométrique et donc

$$u_i^\varepsilon - y^0(t_i) = (1 - 150h)^i(u_0^\varepsilon - y^0(t_0)) = (1 - 150h)^i\varepsilon,\ i = 0, \ldots, n$$

Pour $n = 50$ et $\varepsilon = 0.01$, on obtient $u_n^\varepsilon - y^0(t_n) = (1 - 150h)^n\varepsilon = (-2)^{50} \times 0.01 \approx 1.126 \times 10^{13}$.
Pour obtenir une majoration de la suite géométrique $u_i^\varepsilon - y^0(t_i)$, il suffit que la raison de la suite reste en valeur absolue inférieure à 1, soit $|1 - 150h| \leqslant 1$ ou encore $0 < h \leqslant \dfrac{1}{75}$. Dans ce cas $|u_i^\varepsilon - y^0(t_i)| \leqslant \varepsilon$.

f1.m

```
function z=f1(t,y);
z=-150*y+49-150*t;
```

RK2.m

```
a=0;b=1;
n=100
h=(b-a)/n;
t=a:h:b;
solex=1/3-t;
epsilon=0.01;
u(1)=1/3+epsilon;
for i=1:n
 u(i+1)=u(i)+h/2*(f1(t(i),u(i))+f1(t(i)+h,u(i)+h*f1(t(i),u(i))));
end;
errmax=norm(u-solex,inf)
```

resol1.m

```
[t,u]=ode23('f',[0,1],1/3+1e-2);
disp('Nombre␣d''elements␣de␣t␣:␣')
length(t)
erreur=norm(u-1/3+t,inf)
```

## 8.2 Équation linéaire du second ordre

**1.** Si on pose $Y(x) = \begin{pmatrix} y(x) \\ y'(x) \end{pmatrix} = \begin{pmatrix} y_1(x) \\ y_2(x) \end{pmatrix}$, alors

$$y_1'(x) = y'(x) = y_2(x) \text{ et } y_2'(x) = y''(x) = -ay'(x) - by(x) + f(x) = -ay_2(x) - by_1(x) + f(x)$$

soit $Y'(x) = \begin{pmatrix} 0 & 1 \\ -b & -a \end{pmatrix} Y(x) + \begin{pmatrix} 0 \\ f(x) \end{pmatrix}$. On a donc obtenu $Y'(x) = AY(x) + F(x)$. Les conditions initiales s'écrivent $Y(x_0) = \eta = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$.

**2.** Si $M \in \mathbb{R}^{n\times n}$, alors pour une norme matricielle subordonnée quelconque, nous avons $\|M^n\| \leqslant \|M\|^n$ ce qui explique que la série $\displaystyle\sum_{k=0}^{+\infty} \frac{x^k M^k}{k!}$ soit normalement convergente sur un intervalle borné.

Dans le cas où $M = PDP^{-1}$, alors $M^k = PD^kP^{-1}$ et donc $e^{xM} = P\sum_{k=0}^{+\infty}\frac{x^kD^k}{k!}P^{-1}$. Si $D$ est diagonale $D = \begin{pmatrix}\lambda_1 & & 0\\ & \ddots & \\ 0 & & \lambda_n\end{pmatrix}$, alors $D^k = \begin{pmatrix}\lambda_1^k & & 0\\ & \ddots & \\ 0 & & \lambda_n^k\end{pmatrix}$ et donc

$$e^{xD} = \sum_{k=0}^{+\infty}\frac{x^kD^k}{k!} = \begin{pmatrix}e^{x\lambda_1} & & 0\\ & \ddots & \\ 0 & & e^{x\lambda_n}\end{pmatrix} \text{ si bien que } e^{xM} = Pe^{xD}P^{-1} = P\begin{pmatrix}e^{x\lambda_1} & & 0\\ & \ddots & \\ 0 & & e^{x\lambda_n}\end{pmatrix}P^{-1}.$$

De même que pour la série initiale, la série des dérivées de terme général $\frac{x^{k-1}M^k}{(k-1)!}$ converge normalement sur tout intervalle borné. On peut donc dériver sous le signe $\sum$ et en simplifiant, on obtient $\frac{e^{xM}}{dx} = Me^{xM} = e^{xM}M$.

$$\begin{aligned}
e^{xM}\times e^{x'M} &= \sum_{i=0}^{+\infty}\frac{x^iM^i}{i!}\times\sum_{j=0}^{+\infty}\frac{x'^jM^j}{j!} = \sum_{i=0}^{+\infty}\sum_{j=0}^{+\infty}\frac{x^ix'^j}{i!j!}M^{i+j}\\
&= \sum_{k=0}^{+\infty}\sum_{i=0}^{k}\frac{x^ix'^{k-i}}{i!(k-i)!}M^k = \sum_{k=0}^{+\infty}\frac{1}{k!}\left(\sum_{i=0}^{k}\frac{k!}{i!(k-i)!}x^ix'^{k-i}\right)M^k\\
&= \sum_{k=0}^{+\infty}\frac{1}{k!}(x+x')^kM^k = e^{(x+x')M}
\end{aligned}$$

Enfin en prenant $x' = -x$ et en remarquant que $e^{0M} = Id$, on obtient $(e^{xM})^{-1} = e^{-xM}$.

**3.** Soit $Y$ une solution de $Y'(x) = AY(x)$, puis que $e^{xA}$ est inversible, on peut définir $Z(x) = e^{-xA}Y(x)$. Alors $Z'(x) = -e^{-xA}AY(x) + e^{-xA}Y'(x) = 0$. Donc $Z$ est une fonction constante égale à $\lambda\in\mathbb{R}^2$, alors $Y(x) = e^{xA}\lambda$.

**4.** Si on cherche une solution de $(\mathcal{P}')$ sous la forme $Y(x) = e^{xA}\lambda(x)$, il vient $\lambda'(x) = e^{-xA}F(x)$. En intégrant entre $x_0$ et $x$, on obtient $\lambda(x) = \lambda_0 + \int_{x_0}^{x}e^{-tA}F(t)dt$, d'où

$$Y(x) = e^{xA}\left(\lambda_0 + \int_{x_0}^{x}e^{-tA}F(t)dt\right) = e^{xA}\lambda_0 + \int_{x_0}^{x}e^{(x-t)A}F(t)dt.$$

Enfin en utilisant la condition $Y(x_0) = \eta$, la solution est donnée par

$$Y(x) = e^{(x-x_0)A}\eta + \int_{x_0}^{x}e^{(x-t)A}F(t)dt.$$

## 8.3 Erreur dans la méthode de Runge-Kutta

f2.m

```
function z=f2(t,y);
z(1)=y(2);
z(2)=-y(1);
z=z';
```

### Ce qu'il faut retenir de cet exercice

Par défaut le tableau $z$ est écrit horizontalement même si le tableau $y$ est vertical.

RK4.m

```
function [t,U]=RK4(fichier,n,eta,interv)
h=(interv(2)-interv(1))/n;
t=interv(1):h:interv(2);
U(:,1)=eta;
for i=1:n
    U1=U(:,i);
    K1=feval(fichier,t(i),U1);
    U2=U1+h/2*K1;
    K2=feval(fichier,t(i)+h/2,U2);
    U3=U1+h/2*K2;
    K3=feval(fichier,t(i)+h/2,U3);
    U4=U1+h*K3;
    K4=feval(fichier,t(i)+h,U4);
    U(:,i+1)=U1+h/6*(K1+2*K2+2*K3+K4);
end;
```

### Ce qu'il faut retenir de cet exercice

On crée une fonction RK4 qui est une fonction « bibliothèque », réutilisable pour un autre système différentiel du premier ordre.

erreur1.m

```
interv=[0 1];
fichier='f2';
eta=[0;1];
n=3
[t,U]=RK4(fichier,n,eta,interv);
err=norm(U(1,:)-sin(t),inf)
```

`erreur2.m`

```
interv=[0 1];
fichier='f2';
eta=[0;1];
tn=[7 10 15 20 35 50]
for j=1:length(tn)
    [t,U]=RK4(fichier,tn(j),eta,interv);
    taberr(j)=norm(U(1,:)-sin(t),inf);
end
taberr
plot(log(tn),log(taberr))
xlabel('log(n)','Fontsize',14)
ylabel('log(erreur)','Fontsize',14)
disp('coefficients de la droite de regression')
a=polyfit(log(tn),log(taberr),1)
title(['pente de la droite : ',num2str(a(1))],'Fontsize',18)
```

## 8.4 Système différentiel

**1.** La série des dérivées $\sum_{k=1}^{\infty} A^k \frac{1}{(k-1)!} t^{k-1} = Ae^{At}$ converge normalement sur tout intervalle $[a,b]$ ; on en déduit que $\frac{de^{At}}{dt} = Ae^{At}$. De plus $e^{A0} = Id$. Si $yex = e^{At}\eta$, on obtient $yex'(t) = Ayex(t)$ et $yex(0) = \eta$ donc $yex$ est solution de $\mathcal{S}$. L'unicité résulte du théorème sur les équations différentielles linéaires.

**2.** Par récurrence, $A^k = \begin{pmatrix} (-1)^k & (-1)^{k+1}k \\ 0 & (-1)^k \end{pmatrix}$, donc

$$yex(t) = e^{At}\eta = \left[\sum_{k=0}^{+\infty} \begin{pmatrix} \frac{(-t)^k}{k!} & -\frac{k(-t)^k}{k!} \\ 0 & \frac{(-t)^k}{k!} \end{pmatrix}\right] \begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} (\alpha+\beta t)e^{-t} \\ \beta e^{-t} \end{pmatrix}.$$

**3...5** `SD123.m`

```
a=0;b=1;
n=15
h=(b-a)/n;
t=a:h:b;
eta=[1;2];
A=[-1 1;0 -1];
B=eye(2)+h*A+h^2*A^2/2
U(:,1)=eta;
for i=1:n
    U(:,i+1)=B*U(:,i);
end;
U(:,6)
yex=solex(eta,t);
plot(t,U(1,:)./U(2,:),'.',t,yex(1,:)./yex(2,:))
```

solex.m

```
function y=solex(eta,t);
y(1,:)=(eta(1)+eta(2)*t).*exp(-t);
y(2,:)=eta(2)*exp(-t);
```

**6 et 7** SD5.m

```
clear
a=0;b=1;
eta=[1;2];
A=[-1 1;0 -1];
U(:,1)=eta;
tn=[5 10 20 50 100 200]
for i=1:length(tn)
    n=tn(i);
    h=(b-a)/n;
    B=eye(2)+h*A+h^2*A^2/2;
    t=a:h:b;
    for j=1:n
        U(:,j+1)=B*U(:,j);
    end;
    yex=solex(eta,t);
    err1=U-yex;
    err2=sqrt(err1(1,:).^2+err1(2,:).^2);
    taberr(i)=max(err2);
end
taberr
plot(log(tn),log(taberr))
a=polyfit(log(tn),log(taberr),1);
xlabel('log(n)','Fontsize',14)
ylabel('log(erreur)','Fontsize',14)
title(['Pente de la droite : ',num2str(a(1))],'Fontsize',18)
```

L'étude numérique conduit à penser que la méthode est d'ordre 2 puisque $\log(erreur) = -2\log(n) + c_1$ donne $erreur = \frac{c_2}{n^2}$.

**8.** Partant de $y''(t) = -ty'(t) - y(t)$, en posant $Y = \begin{pmatrix} y \\ y' \end{pmatrix}$, l'équation se transforme en système $Y'(t) = F(t, Y(t))$ où $F(t,U) = A(t)U$ et $A(t) = \begin{pmatrix} 0 & 1 \\ -1 & -t \end{pmatrix}$. On a alors

$$\begin{aligned} & U_{j+1} = U_j + hf\left(t_j + \frac{h}{2}, U_j + \frac{h}{2} f(t_j, U_j)\right) \\ \Leftrightarrow \quad & U_{j+1} = \left(Id + hA\left(t_j + \frac{h}{2}\right) + \frac{h^2}{2} A\left(t_j + \frac{h}{2}\right) A(t_j)\right) U_j. \end{aligned}$$

Les conditions initiales $y(0) = 1$ et $y'(0) = 0$ donne $Y(0) = U_0 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$.

```
function A=matA(t);
A=[0 1;-1 -t];
```

SD6.m

```
a=0;b=1;
n=5
h=(b-a)/n;
t=a:h:b;
eta=[1;0];
U(:,1)=eta;
for j=1:n
    A1=matA(t(j));
    A2=matA(t(j)+h/2);
    B=eye(2)+h*A2+h^2/2*A2*A1;
    U(:,j+1)=B*U(:,j);
end;
U(:,6)
yex=exp(-t.^2/2);
plot(t,U(1,:),'.',t,yex)
```

**9.** SD7.m

```
clear
a=0;b=1;
eta=[1;0];
A=[-1 1;0 -1];
U(:,1)=eta;
tn=[5 10 20 50 100 200]
for i=1:length(tn)
    n=tn(i);
    h=(b-a)/n;
    t=a:h:b;
    for j=1:n
        A1=matA(t(j));
        A2=matA(t(j)+h/2);
        B=eye(2)+h*A2+h^2/2*A2*A1;
        U(:,j+1)=B*U(:,j);
    end;
    yex=exp(-t.^2/2);
    taberr(i)=norm(U(1,:)-yex,inf);
end
taberr
plot(log(tn),log(taberr))
a=polyfit(log(tn),log(taberr),1);
xlabel('log(n)','Fontsize',14)
ylabel('log(erreur)','Fontsize',14)
title(['Pente␣de␣la␣droite␣:␣',num2str(a(1))],'Fontsize',18)
```

L'erreur est encore en $1/n^2$.

## 8.5 Méthode de tir

Comme précédemment, l'équation $y''(t) = 1 - 2t\cos(t) - y(t) + ty'(t)$ se transforme en système $Y'(t) = f(t, Y(t))$ où $Y(t) = \begin{pmatrix} y(t) \\ y'(t) \end{pmatrix}$ et $f(t, U) = \begin{pmatrix} U_2 \\ 1 - 2t\cos t - U_1 + tU_2 \end{pmatrix}$.

f.m

```
function Z=f(t,Y);
Z(1)=Y(2);
Z(2)=1-2*t*cos(t)-Y(1)+t*Y(2);
Z=Z';
```

eul.m

```
function [t,U]=eul(fichier,n,eta,interv)
h=(interv(2)-interv(1))/n;
t=interv(1):h:interv(2);
U(:,1)=eta;
for j=1:n
    U(:,j+1)=U(:,j)+h*feval(fichier,t(j),U(:,j));
end;
```

tir2.m

```
tn=[5 10 20 50 100 200]
alpha=2*(1+sin(1));
a=0;b=1;
for i=1:length(tn)
    n=tn(i);
    [t,U1]=eul('f',n,[1;0],[0,1]);
    [t,U2]=eul('f',n,[1;1],[0,1]);
    lambda=(alpha-U2(1,n+1))/(U1(1,n+1)-U2(1,n+1));
    U0=lambda*U1+(1-lambda)*U2;
    yex=t+2*sin(t)+1;
    taberr(i)=norm(U0(1,:)-yex,inf);
end
taberr
plot(log(tn),log(taberr))
xlabel('log(n)','Fontsize',14)
ylabel('log(erreur)','Fontsize',14)
disp('coefficients de la droite de regression')
a=polyfit(log(tn),log(taberr),1)
title(['pente de la droite : ',num2str(a(1))],'Fontsize',18)
```

Numériquement, on retrouve donc une méthode d'ordre 1.

Pour la méthode de Runge-Kutta, on remplace simplement les appels à `eul` par des appels à `RK4` dont le corrigé a été donné précédemment. Numériquement, on retrouve alors une méthode d'ordre 4.

# 9 Méthodes multipas

Dans le chapitre précédent, nous avons étudié des méthodes numériques à un pas qui permettent de trouver des solutions approchées d'une équation différentielle en passant d'une approximation $u_i$ à un instant $t_i$ à une approximation $u_{i+1}$ à un instant $t_{i+1}$. Dans les méthodes multipas, on utilise non seulement $u_i$, mais aussi les approximations précédentes, $u_{i-1}$, $u_{i-2}$... Par exemple, en reprenant les notations du chapitre précédent, la méthode explicite d'Adams-Bashforth à pas constant $h$ :

$$u_i = u_{i-1} + \frac{h}{24}(55 f_{i-1} - 59 f_{i-2} + 37 f_{i-3} - 9 f_{i-4}) \quad \text{où} \quad f_j = f(t_j, u_j).$$

L'initialisation est alors un peu plus compliquée, puisqu'il faut aussi définir les premières valeurs par une autre méthode. Une erreur conséquente pour ces premières valeurs fera perdre tout l'intérêt d'une méthode précise ensuite. Le premier exercice permet d'illustrer l'importance de cette initialisation.

Dans les méthodes multipas, nous retrouvons les notions de stabilité, d'erreur de consistance, d'ordre... Plutôt que de redéfinir ces notions, nous détaillons une méthode multipas en montrant comment elle s'obtient à partir d'une méthode d'intégration approchée, puis nous montrons la stabilité, nous évaluons l'ordre de la méthode pour conclure à la convergence en précisant la vitesse de convergence. L'étude numérique permet de confirmer que cette majoration est optimale.

## ÉNONCÉS DES EXERCICES

### 9.1 Conditions initiales

**1.** On considère le problème

$$\begin{cases} y'(t) = -y(t), \ t \in [0, +\infty[ \\ y(0) = 1 \end{cases}$$

dont la solution unique est donnée par $y(t) = e^{-t}$. Soient $h$ un pas de temps donné et $t_i = ih$ pour $i = 0, 1, \ldots$ On détermine une solution approchée $u_i$ en $t_i$ par la méthode du point milieu, i.e.

$$\begin{cases} u_0 = 1 \\ u_1 \text{ à fixer} \\ u_{i+1} = u_{i-1} - 2hu_i, \ i = 2, \ldots, n-1. \end{cases}$$

– **1er cas :** On prend comme donnée $u_1 = -h + \sqrt{1+h^2}$. Donner la formule de $u_i$ puis déterminer $\lim\limits_{i \to \infty} u_i$.

– **2e cas :** On suppose maintenant que la donnée initiale $u_1$ est estimée par la méthode d'Euler explicite et donc $u_1 = 1 - h$. Donner la formule de $u_i$ et déterminer $\lim_{i\to\infty} u_i$.

*On constate donc que sur un intervalle de temps très grand le choix de $u_1$ est crucial.*

**2.** Soit $\beta > 0$ et soit le problème de Cauchy suivant :

$$(\mathcal{P}) \begin{cases} y''(t) = -\beta^2 y(t) \\ y(0) = \lambda, y'(0) = \mu \end{cases}$$

(a) Donner la solution exacte du problème $(\mathcal{P})$.

(b) Soient $h$ un pas de temps donné et $t_i = ih$. On pose

$$(\mathcal{P}_h) \begin{cases} \dfrac{u_{k+1} - 2u_k + u_{k-1}}{h^2} = -\beta^2 u_k,\ k = 1, \dots \\ u_0 = \lambda, \\ u_1 = \lambda + h\,\mu - \frac{1}{2} h^2 \beta^2 \lambda \end{cases}$$

*Cette approximation de la dérivée seconde sera étudiée dans le chapitre sur les différences finies.*

Montrer que $(\mathcal{P}_h)$ peut s'écrire sous la forme

$$(\mathcal{P}_h) \begin{cases} u_{k+1} = 2\alpha u_k - u_{k-1},\ k \geqslant 1 \\ u_0 = \lambda, \\ u_1 = \alpha\lambda + h\mu \end{cases}$$

(c) Déterminer $a$ et $b$ pour que la suite $(u_k)$ s'écrive sous la forme $u_k = a\, r_1^k + b\, r_2^k$, pour $k = 0, 1, 2 \dots$ où $r_1$ et $r_2$ sont solutions de $r^2 - 2\alpha r + 1 = 0$.

(d) Sous quelle condition la solution $u$ est-elle oscillante ?

## 9.2 Intégration numérique

Soit $\phi \in \mathcal{C}^3(\mathbb{R}, \mathbb{R})$ et $x_0 \in \mathbb{R}$.

**1.** Pour $h > 0$, montrer qu'il existe un unique polynôme $q \in \mathbb{P}_2$ vérifiant :

$$q(x_0) = \phi(x_0),\ q'(x_0) = \phi'(x_0),\ q(x_0 - 2h) = \phi(x_0 - 2h).$$

**2.** Déterminer $\displaystyle\int_{x_0-2h}^{x_0+h} q(x)dx$ en fonction des valeurs de $\phi$.

**3.** On rappelle que l'erreur est donnée par $\phi(x) - q(x) = \dfrac{1}{3!}\pi_3(x)\phi^{(3)}(\xi_x)$ où $\xi_x \in I$, plus petit intervalle fermé contenant $x, x_0$ et $x_0 - 2h$ et $\pi_3(x) = (x - x_0)^2(x - x_0 + 2h)$ ; il s'agit de l'erreur d'approximation dans l'interpolation d'Hermite (cf [6] par exemple).

En déduire que si $M_3 = \max_{t\in[x_0-2h,x_0+h]} |\phi^{(3)}(t)|$, alors

$$\left| \int_{x_0-2h}^{x_0+h} \phi(x)dx - \frac{9h}{4}\phi(x_0) - \frac{3h}{4}\phi(x_0-2h) \right| \leqslant \frac{3h^4}{8} M_3. \tag{9.1}$$

## 9.3 Une méthode multipas

Soit $f \in \mathcal{C}^3([t_0, t_0+T], \mathbb{R})$ et $\eta \in \mathbb{R}$. On considère l'équation différentielle

$$(P) \begin{cases} y'(t) = f(t, y(t)), t \in [t_0, t_0+T] \\ y(t_0) = \eta \end{cases}$$

et on suppose que $f$ vérifie la condition de Lipschitz :

$$\exists L > 0, \forall t \in [t_0, t_0+T], \forall y, z \in \mathbb{R}, \ |f(t,y) - f(t,z)| \leqslant L|y-z|.$$

Pour approcher la solution $y(\cdot)$ de $(P)$, on subdivise l'intervalle $[t_0, t_0+T]$ avec un pas constant $h = T/n$ et on pose $t_i = t_0 + ih$, pour $i = 0, \ldots, n$. $y(t_i)$ est approchée par $u_i$ définie de la façon suivante :
Pour l'initialisation

$$u_0 = \eta, \ x = u_0 + hf(t_0, u_0), \ u_2 = u_0 + 2hf(t_1, x), \ u_1 = \frac{u_o}{4} + \frac{x}{2} + \frac{u_2}{4}.$$

Ensuite pour $i = 2$ à $n-1$ :

$$u_{i+1} = u_{i-2} + \frac{9h}{4} f(t_i, u_i) + \frac{3h}{4} f(t_{i-2}, u_{i-2}).$$

L'étude des erreurs commises pour les premiers termes est assez technique. On peut passer cette première partie dans un premier temps et se contenter de constater que les majorations entre solution exacte et solution approchée sont en $O(h^3)$.

**1.** Nous allons démontrer successivement que si $M_i = \max_{t\in[t_0,t_0+T]} |y^{(i)}(t)|$,

$$|y(t_1) - x| \leqslant \frac{M_2}{2} h^2, \tag{9.2}$$

$$|y(t_2) - u_2| \leqslant \left( \frac{M_3}{3} + LM_2 \right) h^3, \tag{9.3}$$

$$|y(t_1) - u_1| \leqslant \left( \frac{M_3}{12} + \frac{LM_2}{4} \right) h^3. \tag{9.4}$$

(a) Montrer que $y(t_1) - x = \displaystyle\int_{t_0}^{t_1} \big(y'(t) - y'(t_0)\big)dt$.

(b) En déduire (9.2)

(c) Montrer que $y(t_2) - u_2 = \displaystyle\int_{t_0}^{t_2} \big(y'(t) - y'(t_1) + y'(t_1) - f(t_1, x)\big)dt$.

(d) En utilisant un développement de Taylor de $y'(t)$ en $t_1$, montrer que

$$\left| \int_{t_0}^{t_2} \big(y'(t) - y'(t_1)\big) dt \right| \leqslant \frac{M_3}{3} h^3.$$

(e) Sachant que $y'(t_1) = f(t_1, y(t_1))$, montrer que

$$|y'(t_1) - f(t_1, x)| \leqslant L M_2 \frac{h^2}{2}. \tag{9.5}$$

(f) En déduire (9.3)

(g) Montrer que $|y(t_1) - u_1| = \left| \int_{t_0}^{t_1} \left( y'(t) - \frac{1}{2} y'(t_0) - \frac{1}{2} y'(t_1) + \frac{1}{2} y'(t_1) - \frac{1}{2} f(t_1, x) \right) dt \right|$.

(h) À l'aide la majoration obtenue pour la méthode des trapèzes (5.1), montrer que

$$\left| \int_{t_0}^{t_1} y'(t) dt - h \frac{y'(t_0) + y'(t_1)}{2} \right| \leqslant \frac{M_3}{12} h^3.$$

(i) En utilisant à nouveau (9.5), montrer (9.4).

**2.** Soit $\varepsilon_i = y(t_{i+1}) - y(t_{i-2}) - \frac{9}{4} h f(t_i, y(t_i)) - \frac{3}{4} h f(t_{i-2}, y(t_{i-2}))$, l'erreur de consistance est $\sum_{i=0} |\varepsilon_i\|$ (cf chapitre 8). Majorer $|\varepsilon_i|$ en fonction de $h$ et de $M_4$. Quel est l'ordre de la méthode ?

**3.** On considère la suite $(v_i)$ vérifiant le schéma perturbé

$$v_{i+1} = v_{i-2} + \frac{9h}{4} f(t_i, v_i) + \frac{3h}{4} f(t_{i-2}, v_{i-2}) + \mu_i, 2 \leqslant i \leqslant n-1$$

et on pose $\theta_i = \max\big(|u_i - v_i|, |u_{i-1} - v_{i-1}|, |u_{i-2} - v_{i-2}|\big)$.
Montrer que pour $i \geqslant 2$, $|u_{i+1} - v_{i+1}| \leqslant (1 + 3Lh)\, \theta_i + |\mu_i|$ et en déduire une majoration de $\theta_{i+1}$ en fonction de $\theta_i$.

**4.** Démontrer le lemme suivant (Lemme de Gronwall) :
*Soient $(\alpha_i)$ et $(\beta_i)$ 2 suites de réels positifs vérifiant $\alpha_{i+1} \leqslant (1 + hK)\alpha_i + \beta_i$ pour $i \geqslant 0$, alors*

$$\forall i \geqslant 0,\ \alpha_i \leqslant e^{K(t_i - t_0)} \alpha_0 + \sum_{j=0}^{i-1} e^{K(t_i - t_{j+1})} \beta_j.$$

On pourra le démontrer par récurrence et remarquer que $\forall x \in \mathbb{R}_+,\ 1 + x \leqslant e^x$.

**5.** Déduire de 3) et 4) une majoration de $|u_i - v_i|$ en fonction de

$$|u_0 - v_0|, |u_1 - v_1|, |u_2 - v_2|, |\mu_2|, ..., |\mu_{i-1}|.$$

En déduire que la méthode est stable (cf chapitre 8).

**6.** Déterminer une majoration de l'erreur $|y(t_i) - u_i|$.

## 9.4 Programmation

Étant donné un réel $\eta$ et une fonction $f \in \mathcal{C}^3([t_0, t_0 + T] \times \mathbb{R}; \mathbb{R})$ on considère l'équation différentielle à condition initiale

$$\begin{cases} y'(t) = f(t, y(t)) \quad \text{pour } t \in [t_0, t_0 + T] \\ y(t_0) = \eta. \end{cases}$$

On utilise un pas constant $h = T/N$ et on pose $t_i = t_0 + ih$, $0 \leqslant i \leqslant N$. Le schéma numérique est le suivant :

$$u_0 = \eta,\ x = u_0 + hf(t_0, u_0),\ u_2 = u_0 + 2hf(t_1, x),\ u_1 = \frac{u_0}{4} + \frac{x}{2} + \frac{u_2}{4},$$

puis pour $i = 2, ..., N - 1$,

$$u_{i+1} = u_{i-2} + \frac{9h}{4} f(t_i, u_i) + \frac{3h}{4} f(t_{i-2}, u_{i-2}),$$

où $u_i$ désigne une approximation de $y(t_i)$.

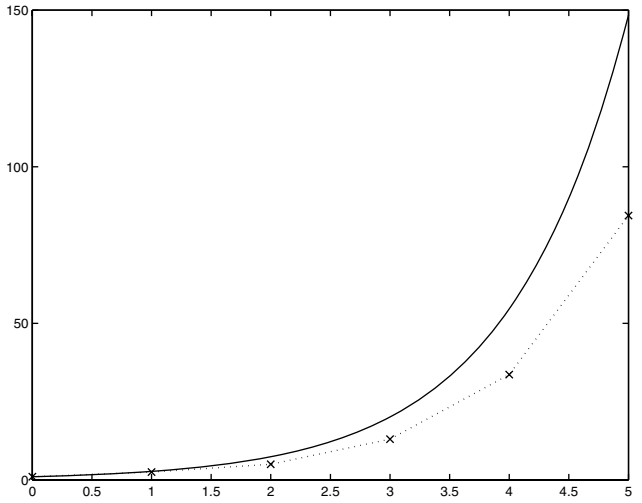
**1.** Écrire une fonction `function [t,u] =multip(nomf,t0,T,N,eta)` qui met en oeuvre ce schéma et calcule la solution approchée $u$ pour la fonction $f$ ; $u$ est un vecteur de composantes $u(i) = u_{i-1}$ correspondant aux instants $t(i) = t_{i-1}$ pour $i = 1, ..., N + 1$, `nomf` est le nom du fichier contenant la fonction $f$ qui définit l'équation différentielle et qui doit être écrite dans le fichier `f.m`. Son en-tête est de la forme `function z=f(t,y)`. Dans `multip`, l'évaluation de cette fonction se fait simplement par `feval(nomf,t,y)`. Écrire le schéma dans les 2 cas suivants :

- **cas 1 :** $y'(t) = y(t)$, $\eta = 1$, $t_0 = 0$, $T = 5$.
- **cas 2 :** $y'(t) = (-y(t)t^2 - y^2(t) + 2t)/(1 - t^3)$, $\eta = 1$, $t_0 = 0$, $T = 0.99$, solution : $y(t) = (t^2 + 1)/(t + 1)$.

**2.** Créer deux fichiers `f1.m, f2.m` qui pour un couple de réels $(t, y)$ calcule $z = f(t, y)$ dans les deux cas proposés.

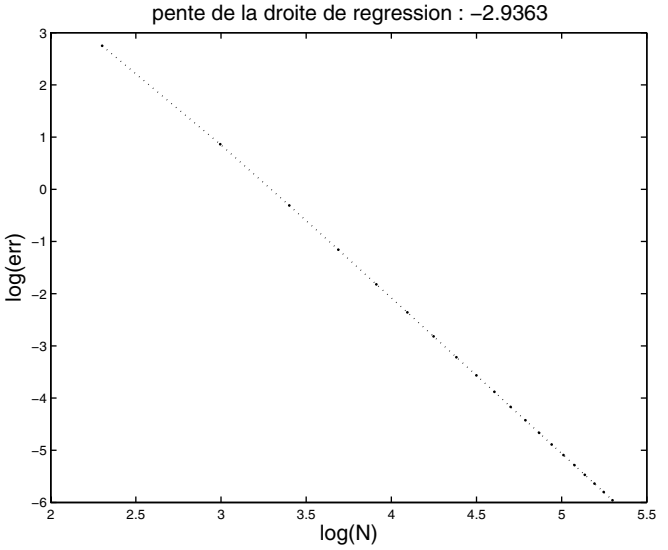
**3.** Écrire un programme permettant de tester le schéma en traçant sur le même graphique la solution exacte et son approximation. Sauvegarde sous `eqdm1.m`. On prendra quelques exemples $N = 5, 10, 20, 30$ dans les deux cas. Test $N = 5$ dans le 1er cas :

```
>> eqdm1
t =
     0     1     2     3     4     5

u =
    1.0000    2.5000    5.0000   13.0000   33.6250   84.4063
```




On pourra aussi tester le cas 2 avec $N = 20$ et $T = 1.23147$. Que se passe-t-il ?

**4.** Dans le cas 1, écrire un programme permettant de tracer, la courbe donnant $\log(err)$ en fonction de $\log(h)$, où $err = \max_{0 \leqslant n \leqslant N} |y(t_n) - u_n|$. On fera varier $N$ de 10 en 10 dans l'intervalle $[10, 200]$ en créant un tableau $tabN$ et un tableau $taberr$. Sauvegarde sous `eqdm2.m`.




Cette courbe permet de mesurer l'ordre du schéma. Y a-t-il concordance avec la prévision théorique ?

**5.** Les équations précédentes sont des équations scalaires. Or le schéma est aussi applicable si $f \in \mathcal{C}^3\left([t_0, t_0 + T] \times \mathbb{R}^d, \mathbb{R}^d\right)$. Ainsi, on souhaite résoudre maintenant l'équation du second

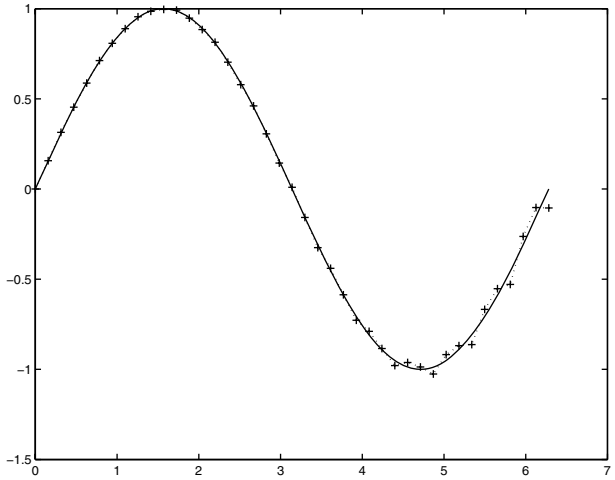
ordre

$$\begin{cases} y''(t) = -y(t),\ t \in [0, 2\pi] \\ y(0) = 0, \\ y'(0) = 1 \end{cases}$$

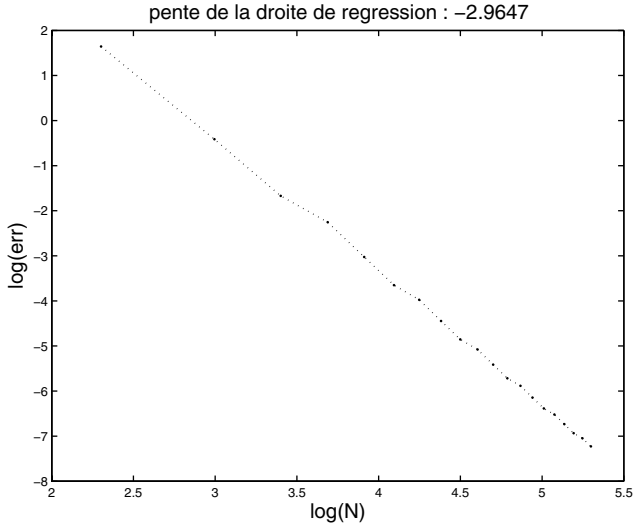
La méthode multipas permet de résoudre un système d'équations différentielles d'ordre 1. Il faut donc au préalable réécrire l'équation de manière à se ramener à un système d'ordre 1 (cf chapitre précédent). Pour prendre en compte des opérandes vectoriels, à savoir *eta* et $y$, le fichier `multip` doit être légèrement modifié, par exemple, remplacer `y(i)` par `y(i, :)`...

Tracer comme précédemment la solution exacte et la solution approchée pour $N = 20, 60, 100$. Comparer éventuellement avec la fonction `ode23` (voir aide en ligne) de `Matlab`. Tracer enfin la courbe donnant $taberr$ en fonction de $tabN$ pour $N = 10 : 10 : 200$.

```
>> eqdm3
```




```
>> eqdm4
```



## DU MAL À DÉMARRER ?

### 9.1 Conditions initiales

L'ensemble des suites récurrentes $(u_i)$ telles que $u_{i+1} = u_{i-1} - 2hu_i$ est un espace vectoriel de dimension 2. On en cherche des éléments de la forme $u_n = r^n$.

### 9.2 Intégration numérique

**1.** On peut utiliser la base $\{1, (x - x_0), (x - x_0)^2\}$.

**3.** $\pi_3(x)$ garde un signe constant sur $[x_0 - 2h, x_0 + h]$.

### 9.3 Une méthode multipas

**1.** (a) $y(t_1) = u_0 + \displaystyle\int_{t_0}^{t_1} y'(t)dt$ et $x = u_0 + \displaystyle\int_{t_0}^{t_1} y'(t_0)dt$.

(b) Utiliser le théorème des accroissements finis.

(c) $y(t_2) = u_0 + \displaystyle\int_{t_0}^{t_2} y'(t)dt$ et $u_2 = u_0 + \displaystyle\int_{t_0}^{t_2} f(t_1, x)dt$.

**2.** $y(t_{i+1}) - y(t_{i-2}) = \displaystyle\int_{t_{i-1}}^{t_{i+1}} y'(t)dt$ et reprendre le paragraphe précédent.

**3.** Utiliser l'hypothèse de Lipschicité de $f$.

## CORRIGÉS DES EXERCICES

### 9.1 Conditions initiales

**1.** Si $h > 0$, l'ensemble des suites récurrentes vérifiant $u_{i+1} = u_{i-1} - 2hu_i$ est un espace vectoriel de dimension 2. Pour en chercher une base, on recherche des solutions de la forme $u_i = r^i$ et on obtient l'équation $r^2 + 2hr - 1 = 0$ dont les 2 racines distinctes sont $r_1 = -h - \sqrt{h^2 + 1}$ et $r_2 = -h + \sqrt{h^2 + 1}$, si bien que ces suites récurrentes sont de la forme $u_i = \lambda r_1^i + \mu r_2^i$.

1er cas : Si on impose $u_0 = 1$ et $u_1 = -h + \sqrt{h^2 + 1} = r_2$, on a alors le système de 2 équations linéaires $\begin{cases} \lambda + \mu = 1 \\ \lambda r_1 + \mu r_2 = r_2 \end{cases}$ dont l'unique solution est $(\lambda, \mu) = (1, 0)$.

Il vient $u_i = (-h + \sqrt{h^2 + 1})^i$. Sachant que $0 < -h + \sqrt{h^2 + 1} < 1$, on obtient que $\lim\limits_{i \to +\infty} u_i = 0$ comme $\lim\limits_{t \to +\infty} y(t) = 0$ où $y$ est la solution exacte de l'équation.

2ème cas : Si on impose $u_0 = 1$ et $u_1 = 1 - h$, on a alors le système de deux équations linéaires $\begin{cases} \lambda + \mu = 1 \\ \lambda r_1 + \mu r_2 = 1 - h \end{cases}$ dont l'unique solution est $(\lambda, \mu) = \left( \dfrac{r_2 - 1 + h}{r_2 - r_1}, \dfrac{1 - h - r_1}{r_2 - r_1} \right)$. Il est facile de montrer que $\lambda \neq 0$ et que $r_1 \leqslant -1$ puisque $h > 0$. Dans ce cas $\lim\limits_{i \to +\infty} |u_i| = +\infty$.

Sachant que $-h + \sqrt{h^2 + 1} = 1 - h + h^2/2 + o(h^2)$. Si $h$ est petit, la différence entre les deux approximations initiales $u_1$ est petite, de l'ordre de $h^2/2$ et pourtant les comportements des deux solutions approchées sont très différents quand $i$ devient grand avec $n$ suffisamment grand.

**2.** Les solutions de $y''(t) = -\beta^2 y(t)$ forment un sous-espace vectoriel de dimension 2 de $\mathcal{C}^2(\mathbb{R})$. Elles sont de la forme $y(t) = y(0)\cos(\beta t) + \dfrac{y'(0)}{\beta}\sin(\beta t)$. Si dans $(\mathcal{P})$, on impose de plus $y(0) = \lambda$ et $y'(0) = \mu$, l'unique solution est alors définie par $y(t) = \lambda\cos(\beta t) + \dfrac{\mu}{\beta}\sin(\beta t)$, fonction périodique.

L'équation $\dfrac{u_{k+1} - 2u_k + u_{k-1}}{h^2} = -\beta^2 u_k$ peut s'écrire $u_{k+1} = 2\alpha u_k - u_{k-1}$ avec $\alpha = 1 - \dfrac{h^2\beta^2}{2}$.

Ajoutons que $u_0 = \lambda$ et $u_1 = \lambda + h\mu - \dfrac{1}{2}h^2\beta^2 = \alpha\lambda + h\mu$. Notons que $\alpha < 1$ puisque $h > 0$.

Comme dans l'exercice précédent, l'ensemble des suites vérifiant $u_{k+1} = 2\alpha u_k - u_{k-1}$ est un sous-espace vectoriel de dimension 2. En recherchant des éléments sous la forme $u_k = r^k$, on trouve 2 solutions distinctes, réelles $r_1 = \alpha + \sqrt{\alpha^2 - 1}$, $r_2 = \alpha - \sqrt{\alpha^2 - 1}$ si $\alpha < -1$ et complexes $r_1 = \alpha + i\sqrt{1 - \alpha^2}$, $r_2 = \alpha - i\sqrt{1 - \alpha^2}$ si $\alpha > -1$ ; enfin on a une racine double quand $\alpha = -1$. Pour $\alpha \neq -1$, on obtient $u_k = ar_1^k + br_2^k$ et pour $\alpha = -1$, $u_k = (a + kb)(-1)^k$. Les coefficients $a$ et $b$ sont déterminés par les conditions sur $u_0$ et $u_1$. Dans le premier cas
$(a, b) = \left( \dfrac{\lambda(r_2 - \alpha) - h\mu}{r_2 - r_1}, \dfrac{\lambda(\alpha - r_1) + h\mu}{r_2 - r_1} \right)$.

Si $r_1$ et $r_2$ sont réelles, la solution n'est pas oscillante. Si par contre $\alpha > -1 \Leftrightarrow h < \frac{2}{\beta}$, alors $|r_1| = 1$ et nous pouvons écrire $r_1 = e^{i\theta}$. On en déduit que $r_2 = e^{-i\theta}$ si bien que la solution $u_k$ peut s'écrire $u_k = a'\cos(\theta k) + b'\sin(\theta k)$ ; elle est oscillante.

## 9.2 Intégration numérique

On choisit la base $\{1, (x - x_0), (x - x_0)^2\}$ de $\mathbb{P}_2$. Bien entendu une autre base est possible mais elle compliquera les calculs. On cherche $q(x) = \alpha 1 + \beta(x - x_0) + \gamma(x - x_0)^2$. Les conditions s'écrivent : $q(x_0) = \phi(x_0) = \alpha$, $q'(x_0) = \phi'(x_0) = \beta$ et $q(x_0 - 2h) = \phi(x_0 - 2h) = \alpha - 2h\beta + 4h^2\gamma$ dont la solution unique est donnée par

$$q(x) = \phi(x_0) + \phi'(x_0)(x - x_0) + \frac{\phi(x_0 - 2h) - \phi(x_0) + 2h\phi'(x_0)}{4h^2}(x - x_0)^2.$$

Alors

$$\begin{aligned}\int_{x_0-2h}^{x_0+h} q(x)dx &= \left[\phi(x_0)(x-x_0)+\phi'(x_0)\frac{(x-x_0)^2}{2}\right.\\ &+ \left.\frac{\phi(x_0-2h)-\phi(x_0)+2h\phi'(x_0)}{4h^2}\frac{(x-x_0)^3}{3}\right]_{x_0-2h}^{x_0+h}\\ &= h\left(\frac{9}{4}\phi(x_0)+\frac{3}{4}\phi(x_0-2h)\right).\end{aligned}$$

Sachant que $\phi(x)-q(x)=\dfrac{1}{3!}\pi_3(x)\phi^{(3)}(\xi_x)$ où $\xi_x \in I$, plus petit intervalle fermé contenant $x, x_0$ et $x_0-2h$ et $\pi_3(x)=(x-x_0)^2(x-x_0+2h)$, on note que $\pi_3$ garde un signe positif sur $[x_0-2h, x_0+h]$ et que $|\phi^{(3)}(\xi_x)| \leqslant M_3$. En intégrant l'erreur, il vient :

$$\begin{aligned}\left|\int_{x_0-2h}^{x_0+h}\phi(x)dx - h\left(\frac{9}{4}\phi(x_0)+\frac{3}{4}\phi(x_0-2h)\right)\right| &= \left|\int_{x_0-2h}^{x_0+h}(\phi(x)-q(x))dx\right|\\ &\leqslant \int_{x_0-2h}^{x_0+h}|\phi(x)-q(x)|\,dx\\ &\leqslant \frac{M_3}{6}\int_{x_0-2h}^{x_0+h}(x-x_0)^2(x-x_0+2h)dx\\ &= \frac{M_3}{6}\left[\frac{(x-x_0)^4}{4}+2h\frac{(x-x_0)^3}{3}\right]_{x_0-2h}^{x_0+h}\\ &= \frac{3}{8}h^4M_3.\end{aligned}$$

## 9.3 Une méthode multipas

**1.** Montrons que $|y(t_1)-x| \leqslant M_2\dfrac{h^2}{2}$.

Puisque $u_0 = y(t_0)$, on obtient $y(t_1) = u_0 + \displaystyle\int_{t_0}^{t_1} y'(t)dt$. Rappelons que $y'(t) = f(t, y(t))$ et que $t_1 - t_0 = h$ donc $x = u_0 + hf(t_0, u_0) = u_0 + hy'(t_0) = u_0 + \displaystyle\int_{t_0}^{t_1} y'(t_0)dt$, si bien que $y(t_1)-x = \displaystyle\int_{t_0}^{t_1}\big(y'(t)-y'(t_0)\big)dt$. Or pour $t \in [t_0, t_1]$, par le théorème des accroissements finis, $y'(t)-y'(t_0) = (t-t_0)y''(\xi)$ où $\xi \in [t_0, t] \subset [t_0, t_0+T]$. $|y''(\xi)|$ se majore par $\max\limits_{t\in[t_0,t_0+T]}|y''(t)|$ que nous notons $M_2$. Il vient alors

$$|y(t_1)-x| \leqslant M_2\int_{t_0}^{t_1}(t-t_0)dt = M_2\left[\frac{(t-t_0)^2}{2}\right]_{t_0}^{t_1} = M_2\frac{h^2}{2}.$$

Montrons ensuite que $|y(t_2) - u_2| \leqslant \left(\frac{M_3}{3} + LM_2\right) h^3$.

D'une part, nous avons $y(t_2) = u_0 + \int_{t_0}^{t_2} y'(t)dt$,

d'autre part $u_2 = u_0 + 2hf(t_1, x) = u_0 + \int_{t_0}^{t_2} f(t_1, x)dt$.

Par différence, nous obtenons

$$y(t_2) - u_2 = \int_{t_0}^{t_2} \big(y'(t) - y'(t_1) + y'(t_1) - f(t_1, x)\big)dt.$$

Nous utilisons un développement de Taylor. Pour $t \in [t_0, t_2]$, il existe $\xi \in ]t_0, t[$ tel que

$$y'(t) = y'(t_1) + (t - t_1)y''(t_1) + \frac{(t - t_1)^2}{2} y^{(3)}(\xi).$$

Et puisque $\int_{t_0}^{t_2} (t - t_1)dt = 0$, on obtient

$$\left|\int_{t_0}^{t_2} (y'(t) - y'(t_1))dt\right| = \left|\int_{t_0}^{t_2} \frac{(t - t_1)^2}{2} y^{(3)}(\xi)dt\right| \leqslant M_3 \left[\frac{(t - t_1)^3}{6}\right]_{t_0}^{t_2} = \frac{M_3}{3}h^3.$$

Notons que $y'(t_1) = f(t_1, y(t_1))$ puis en utilisant la lipschicité de $f$ par rapport à la seconde variable il vient

$$|y'(t_1) - f(t_1, x)| = |f(t_1, y(t_1)) - f(t_1, x)| \leqslant L|y(t_1) - x| \leqslant LM_2 \frac{h^2}{2}.$$

En intégrant ces constantes entre $t_0$ et $t_2$, $\left|\int_{t_0}^{t_2} \big(y'(t_1) - f(t_1, x)\big)dt\right| \leqslant 2hLM_2 \frac{h^2}{2} = LM_2h^3$.

En regroupant les 2 majorations :

$$|y(t_2) - u_2| \leqslant \left|\int_{t_0}^{t_2} y'(t) - y'(t_1)dt\right| + \left|\int_{t_0}^{t_2} y'(t_1) - f(t_1, x)dt\right| \leqslant \left(\frac{M_3}{3} + LM_2\right) h^3.$$

Enfin, montrons que $|y(t_1) - u_1| \leqslant \left(\frac{M_3}{12} + \frac{LM_2}{4}\right) h^3$. Le raisonnement est semblable au précédent en utilisant $y(t_1) = u_0 + \int_{t_0}^{t_1} y'(t)dt$ et $u_1 = \frac{u_0}{4} + \frac{x}{2} + \frac{y_2}{4} = u_0 + \frac{1}{2}hf(t_0, u_0) + \frac{1}{2}hf(t_1, x)$. Sachant que $y'(t_0) = f(t_0, u_0)$ et que $t_1 - t_0 = h$, par différence, nous obtenons

$$\begin{aligned} |y(t_1) - u_1| &= \left|\int_{t_0}^{t_1} \left(y'(t) - \frac{1}{2}y'(t_0) - \frac{1}{2}y'(t_1) + \frac{1}{2}y'(t_1) - \frac{1}{2}f(t_1, x)\right) dt\right| \\ &\leqslant \left|\int_{t_0}^{t_1} y'(t)dt - h\frac{y'(t_0) + y'(t_1)}{2}\right| + \frac{1}{2}\int_{t_0}^{t_1} |y'(t_1) - f(t_1, x)|dt \end{aligned}$$

Le premier terme a été majoré lors de l'étude de la méthode des trapèzes (chapitre 5, (5.1) avec $f = y'$). Nous savons que $\left| \int_{t_0}^{t_1} y'(t)dt - h\frac{y'(t_0)+y'(t_1)}{2} \right| \leqslant \frac{M_3}{12}h^3$. Pour le second, nous réutilisons la majoration précédente $|y'(t_1) - f(t_1, x)| \leqslant LM_2\frac{h^2}{2}$ qu'on intègre entre $t_0$ et $t_1$.
Nous pouvons alors conclure que $|y(t_1) - u_1| \leqslant \left(\frac{M_3}{12} + \frac{LM_2}{4}\right)h^3$.

**2.** Sachant que $\varepsilon_i = y(t_{i+1}) - y(t_{i-2}) - \frac{9}{4}hf(t_i, y(t_i)) - \frac{3}{4}hf(t_{i-2}, y(t_{i-2}))$, il vient immédiatement $\varepsilon_i = \int_{t_{i-2}}^{t_{i+1}} y'(t)dt - \frac{9}{4}hf(t_i, y(t_i)) - \frac{3}{4}hf(t_{i-2}, y(t_{i-2}))$. On reconnaît l'erreur étudiée au paragraphe précédent (9.1) avec $\phi = y'$ donc $|\varepsilon_i| \leqslant \frac{3h^4}{8} \max_{t\in[t_{i-2},t_{i+1}]} \left|y^{(4)}(t)\right| \leqslant \frac{3h^4}{8}M_4$, ce qui nous donne une méthode d'ordre 3.

**3.** Pour $i \geqslant 2$, on a

$$\begin{aligned} v_{i+1} &= v_{i-2} + \frac{9h}{4}f(t_i, v_i) + \frac{3h}{4}f(t_{i-2}, v_{i-2}) + \mu_i \\ u_{i+1} &= u_{i-2} + \frac{9h}{4}f(t_i, u_i) + \frac{3h}{4}f(t_{i-2}, u_{i-2}) \end{aligned}$$

Par différence et en utilisant la lipschicité de $f$, il vient

$$|v_{i+1} - u_{i+1}| \leqslant |v_{i-2} - u_{i-2}| + \frac{9h}{4}L|v_i - u_i| + \frac{3h}{4}L|v_{i-2} - u_{i-2}| + |\mu_i| \leqslant (1 + 3Lh)\theta_i + |\mu_i|,$$

où $\theta_i = \max(|v_{i-2} - u_{i-2}|, |v_{i-1} - u_{i-1}|, |v_i - u_i|)$.
Sachant que $|v_i - u_i|$ et $|v_{i-1} - u_{i-1}|$ peuvent être majorés successivement par $\theta_i$, puis par $(1 + 3Lh)\ \theta_i$ et enfin par $(1 + 3Lh)\ \theta_i + |\mu_i|$, nous en déduisons que pour tout $i \geqslant 2$

$$\theta_{i+1} \leqslant (1 + 3Lh)\ \theta_i + |\mu_i|. \tag{9.6}$$

**4.** Soit $(\alpha_i)$ et $(\beta_i)$ 2 suites de réels positifs vérifiant $\alpha_{i+1} \leqslant (1 + hK)\alpha_i + \beta_i$ pour tout $i$ entier positif. Nous allons montrer par récurrence sur $i$ que

$$\alpha_i \leqslant e^{K(t_i - t_0)}\alpha_0 + \sum_{j=0}^{i-1} e^{K(t_i - t_{j+1})}\beta_j. \tag{9.7}$$

Il est facile de montrer que pour tout $x \leqslant 0$, nous avons $e^x \geqslant 1 + x$ par étude de la fonction différence ou en remarquant que $1 + x$ sont les 2 premiers termes du développement en série entière de $e^x$ qui a des termes positifs. Nous en déduisons $1 + hK \leqslant e^{hK}$.
Pour $i = 0$, nous avons $\alpha_0 \leqslant \alpha_0 e^0 = \alpha_0 e^{t_0 - t_0} + 0$.

Si (9.7) est vérifiée, sachant que $\alpha_{i+1} \leqslant (1+hK)\alpha_i + \beta_i$ et que $t_{i+1} = t_i + h$, il vient

$$\begin{aligned}
\alpha_{i+1} &\leqslant (1+hK)\left[e^{K(t_i-t_0)}\alpha_0 + \sum_{j=0}^{i-1} e^{K(t_i-t_{j+1})}\beta_j\right] + \beta_i \\
&\leqslant e^{hK}\left[e^{K(t_i-t_0)}\alpha_0 + \sum_{j=0}^{i-1} e^{K(t_i-t_{j+1})}\beta_j\right] + \beta_i \\
&\leqslant e^{K(t_{i+1}-t_0)}\alpha_0 + \sum_{j=0}^{i-1} e^{K(t_{i+1}-t_{j+1})}\beta_j + \beta_i \\
&= e^{K(t_{i+1}-t_0)}\alpha_0 + \sum_{j=0}^{n} e^{K(t_{i+1}-t_{j+1})}\beta_j.
\end{aligned}$$

Ce qui conclut la récurrence.

**5.** Posons $\alpha_i = \theta_{i+2}$ pour $0 \leqslant i \leqslant N-2$ et $\beta_i = |\mu_{i+2}|$ pour $0 \leqslant i \leqslant N-3$ . En utilisant (9.7) avec $K = 3L$, l'inégalité (9.6) permet de déduire que $\theta_{i+2} \leqslant e^{K(t_i-t_0)}\theta_2 + \sum_{j=0}^{i-1} e^{K(t_i-t_{j+1})}|\mu_{j+2}|$ pour $i$ variant de 0 à $N-2$. Nous majorons alors $t_i - t_0$ et $t_i - t_{j+1}$ par $T$, longueur de l'intervalle pour obtenir $\theta_{i+2} \leqslant e^{KT}\left(\theta_2 + \sum_{j=0}^{i-1}|\mu_{j+2}|\right)$ ou en translatant la formule, pour $2 \leqslant i \leqslant N$, $\theta_i \leqslant e^{KT}\left(\theta_2 + \sum_{j=2}^{i-1}|\mu_j|\right)$ en rappelant que $\theta_2 = \max(|u_0 - v_0|, |u_1 - v_1|, |u_2 - v_2|)$.

**6.** Si nous choisissons $v_i = y(t_i)$, alors $u_0 = v_0$ si bien que $\theta_2 = \max_{j=1,2}(|u_j - y(t_j)|)$. D'après la première question, on a $\theta_2 = O(h^3)$. De plus, pour $j \geqslant 2$, $\mu_j = \varepsilon_j$ par définition de $\varepsilon_j$, si bien que pour $i \in \{3, \ldots, N\}$, nous avons

$$|y(t_i) - u_i| \leqslant e^{KT}\left(O(h^3) + \sum_{j=3}^{N}\frac{3}{8}h^4 M_4\right) = O(h^3)$$

puisque $N = T/h$. On voit ici l'importance de la cohérence de l'erreur sur les premiers termes $O(h^3)$ avec l'erreur de consistance.

## 9.4 Programmation

eqdm1

```
N=5;
nomf='f1';
t0=0;
T=5;
eta=1;
[t,u]=multip(nomf,t0,T,N,eta)
tt=t0:T/500:t0+T;
yex=exp(tt);
plot(t,u,'rx',tt,yex,'b')
```

```
function[t,u]=multip(nomf,t0,T,N,eta)
h=T/N;
t=[t0:h:t0+T];
u(1)=eta;
x=u(1)+h*feval(nomf,t0,u(1));
u(3)=u(1)+2*h*feval(nomf,t(2),x);
u(2)=u(1)/4+x/2+u(3)/4;
c1=9*h/4;
c2=3*h/4;
for n=3:length(t)-1
 u(n+1)=u(n-2)+c1*feval(nomf,t(n),u(n))+c2
                *feval(nomf,t(n-2),u(n-2));
end
```

Dans le cas où $y'(t) = (-y(t)t^2 - y^2(t) + 2t)/(1 - t^3)$ avec $y(0) = 1$, la solution est donnée par $y(t) = (t^2 + 1)/(t + 1)$ au voisinage de 0. A priori l'équation ne peut être vérifiée en 1, mais la solution, elle peut être prolongée. En choisissant une valeur $T > 1$ et un $N$ qui évite la valeur $t_i = 1$, Matlab effectue des calculs au delà de 1 mais les résultats n'ont plus rien à voir avec la solution...

eqdm2.m, étude d'erreur

```
tabN=10:10:200;
for k=1:length(tabN)
    [t,u]=multip('f1',0,5,tabN(k),1);
    yex=exp(t);
    taberr(k)=norm(u-yex,inf);
end
plot(log(tabN),log(taberr))
xlabel('log(N)','FontSize',16)
ylabel('log(err)','FontSize',16)
a=polyfit(log(tabN),log(taberr),1)
title(['pente␣de␣la␣droite␣de␣regression␣:␣'...
    ,num2str(a(1))],'FontSize',16)
```

On retrouve numériquement que $\log(erreur)$ est de la forme $b - 3\log(N)$, soit une erreur numérique de l'ordre de $C/N^3$.

```
function [t,u]=multipmod(nomf,t0,T,N,eta)
h=T/N;
t=[t0:h:t0+T];
u(:,1)=eta;
x=u(:,1)+h*feval(nomf,t0,u(:,1));
u(:,3)=u(:,1)+2*h*feval(nomf,t(2),x);
u(:,2)=u(:,1)/4+x/2+u(:,3)/4;
c1=9*h/4;
c2=3*h/4;
for n=3:length(t)-1
   u(:,n+1)=u(:,n-2)+c1*feval(nomf,t(n),u(:,n))+...
       c2*feval(nomf,t(n-2),u(:,n-2));
end
```

### Ce qu'il faut retenir de cet exercice

Comme dans l'exercice 8.3, on crée ici une fonction « bibliothèque » qui permettra de résoudre des systèmes différentiels.

`eqdm4.m`, étude d'erreur

```
tabN=10:10:200;
for k=1:length(tabN)
    [t,u]=multipmod('F3',0,2*pi,tabN(k),[0;1]);
    yex=sin(t);
    taberr(k)=norm(u(1,:)-yex,inf);
end
plot(log(tabN),log(taberr))
xlabel('log(N)','FontSize',16)
ylabel('log(err)','FontSize',16)
a=polyfit(log(tabN),log(taberr),1)
title(['pente␣de␣la␣droite␣de␣regression␣:␣'...
    ,num2str(a(1))],'FontSize',16)
```

À nouveau l'erreur est en $O(N^{-3})$.

# 10

# Différences finies en dimension 1

## RAPPEL DE COURS

Les différences finies permettent d'obtenir des approximations des dérivées d'une fonction $f$. Si nous prenons trois réels $x_{i-1}, x_i, x_{i+1}$ tels que $h = x_{i+1} - x_i = x_i - x_{i-1} > 0$, en utilisant des formules de Taylor, il existe $\theta_1$, $\theta_2$ dans $]0, 1[$ et $\theta_3$, $\theta_4$ dans $]-1, 1[$ tels que :

$$f'(x_i) = \frac{f(x_{i+1}) - f(x_i)}{h} - \frac{h}{2} f''(x_i + \theta_1 h) \text{ si } f \in \mathcal{C}^2([x_i, x_{i+1}]) \tag{10.1}$$

$$f'(x_i) = \frac{f(x_i) - f(x_{i-1})}{h} + \frac{h}{2} f''(x_i - \theta_2 h) \text{ si } f \in \mathcal{C}^2([x_{i-1}, x_i]) \tag{10.2}$$

$$f'(x_i) = \frac{f(x_{i+1}) - f(x_{i-1})}{2h} - \frac{h^2}{6} f^{(3)}(x_i + \theta_3 h) \text{ si } f \in \mathcal{C}^3([x_{i-1}, x_i]) \tag{10.3}$$

$$f''(x_i) = \frac{f(x_i) - 2f(x_i) + f(x_{i-1})}{h^2} - \frac{h^2}{12} f^{(4)}(x_i + \theta_4 h) \text{ si } f \in \mathcal{C}^4([x_{i-1}, x_i]) \tag{10.4}$$

**Application :** Nous utiliserons ces approximations pour résoudre le problème différentiel $-u''(x) + c(x)u(x) = f(x)$ pour $a \leqslant x \leqslant b$ avec conditions aux extrémités : $u(a) = \alpha$, $u(b) = \beta$ qui *admet une solution unique* de classe $\mathcal{C}^2$ sur $[a, b]$ dès que les fonctions $f$ et $c$ sont continues avec $c \geqslant 0$ ainsi que d'autres problèmes différentiels avec conditions de bords.

Le premier exercice reprend l'équation $-u''(x) + c(x)u(x) = f(x)$ puis, avec les conditions aux bords, montre l'existence et l'unicité de la solution dans le cas où $c$ est une fonction constante strictement positive. On construit pas à pas un problème approché à l'aide de (10.4) et on montre qu'il admet une solution unique. Enfin, on compare les solutions des problèmes exact et approché en étudiant l'erreur.

Dans l'exercice suivant, l'équation différentielle n'est plus linéaire. On verra comment, en rajoutant une méthode itérative de résolution d'équation, on peut maintenir une erreur du même ordre que dans la cas linéaire.

Le schéma de Numerov montre comment sur le problème initial, on peut améliorer la précision de la méthode approchée par une variation sur les différences finies. L'étude d'erreur se fait numériquement pour cet exemple.

Enfin le dernier exercice propose l'étude d'une équation du type $-u''(x) + c(x)u'(x) = f(x)$ avec conditions aux bords d'un intervalle en utilisant (10.3) et (10.4). On verra comment dans certains cas la solution approchée s'écarte de la solution exacte et comment maintenir une erreur du même ordre que celle trouvée dans le cas initial.

## ÉNONCÉS DES EXERCICES

### 10.1 Flexion d'une poutre

On considère une poutre de longueur 1 étirée aux 2 bouts selon son axe par une force $P$ et soumise à une charge transversale $f$. Le moment fléchissant $u(x)$ au point $x$ est solution du problème

$$(P)\ \begin{cases} -u''(x) + c(x)u(x) = f(x) \text{ pour } 0 \leqslant x \leqslant 1 \\ u(0) = u(1) = 0 \end{cases}$$

où $c(x) = P/EI(x)$, $E$ ťant le module d'Young du matériau et $I(x)$ moment principal d'inertie de la section de la poutre au point $x$. Pour simplifier, dans la suite, *on suppose que* $c(x)$ *est une constante* $c > 0$.

Soit $N$ un entier, on subdivise l'intervalle $[0, 1]$ en segments de longueur $h = 1/(N + 1)$. Soit $x_i = ih, i = 0, N + 1$. On va chercher une solution approchée $(v_0, ..., v_{N+1})$ définie en les $x_i$. Naturellement on prend $v_0 = v_{N+1} = 0$ et on cherche $V_h = (v_1, ..., v_N)^T$.

**1.** Montrer que si $f \in \mathcal{C}^0([0, 1])$, le problème $(P)$ admet une solution unique. On peut faire une démonstration directe ou utiliser l'exercice 8.2.

**2.** Soit $\phi$ une fonction de classe $\mathcal{C}^4$ sur $\mathbb{R}$. Montrer que pour tout $x \in \mathbb{R}$, il existe $\alpha \in ]-1, 1[$ tel que

$$\phi''(x) = \frac{\phi(x - h) - 2\phi(x) + \phi(x + h)}{h^2} + c_1 h^2 \phi^{(4)}(x + \alpha h) \tag{10.5}$$

**3.** Écrire les équations en les $x_i$ en remplaçant $u''(x_i)$ par l'expression précédente.

**4.** En négligeant le terme en $h^2$ et en approchant alors $u(x_i)$ par $v_i$, écrire le problème $(P_h)$ sous la forme

$$A_h V_h = F_h \text{ où } A_h \in \mathbb{R}^{N\times N},\ F_h = (h^2 f(x_1), ..., h^2 f(x_N))^T$$

Nous allons maintenant montrer que $(P_h)$ admet une solution unique puis majorer l'erreur $V_h - U_h$ où $U_h$ est la solution exacte en les $x_i$, $U_h = (u(x_1), ..., u(x_N))^T$.

**5.** On dit qu'un vecteur $V$ *de* $\mathbb{R}^N$ *est positif* (noté $V \geqslant 0$) si tous ses composants sont positifs. De même pour une *matrice positive*.

(a) Montrer que si $A_h V \geqslant 0$ alors $V \geqslant 0$.

(b) En déduire que $A_h$ est inversible et que $A_h^{-1} \geqslant 0$ ; on dit dans ce cas que $A_h$ est une *matrice monotone*.

(c) Montrer que $(P_h)$ admet une solution unique.

**6.** Étude de l'erreur. On suppose que la solution $u$ est dans $\mathcal{C}^4([0,1])$.

Soient $w$ et $\theta$ deux vecteurs tels que $A_h w = \theta$ ; on définit $\psi(x) = \dfrac{x(1-x)}{2h^2}\,\|\theta\|_\infty$ et $\Psi = (\psi(x_1), ..., \psi(x_N))^T$.

(a) Calculer $A_h\Psi$ , montrer que $A_h\Psi \geqslant A_h w$ et $A_h\Psi \geqslant A_h(-w)$.

(b) En déduire que $\|w\|_\infty \leqslant \|\Psi\|_\infty \leqslant \dfrac{1}{8h^2}\,\|\theta\|_\infty$.

(c) Montrer que $A_h(U_h - V_h) = c_1 h^4 (u^{(4)}(x_1'), ..., u^{(4)}(x_N'))^T$.

(d) Majorer alors l'erreur $\|U_h - V_h\|_\infty$ en fonction de $h$ et de $M_4 = \max_{x\in[0,1]} |u^{(4)}(x)|$.

Même si l'erreur sur l'approximation de $u''$ est en $h^2$, il n'était pas évident que l'erreur finale reste de l'ordre de $h^2$ car il y a aussi une résolution de système de dimension $N = 1/h$. Le conditionnement de la matrice $A_h$ n'augmente donc pas avec $N$.

## 10.2 Oscillations d'un pendule

On souhaite résoudre le problème suivant, qui décrit le mouvement d'un pendule oscillant, soumis à une excitation en imposant des points de passage en $a$ et $b$ :

$$(P)\ \begin{cases} u''(t) + \sin(u(t)) = f(t) \text{ pour } t \in [a,b] \\ u(a) = \alpha, u(b) = \beta \end{cases}$$

On construit un problème approché par une méthode de différences finies.Pour cela, l'intervalle $[a,b]$ est subdivisé avec un pas constant $h = (b-a)/(N+1)$. On appelle $t = (t_1, \ldots, t_{N+2})^T$ le vecteur tel que $t_i = a + (i-1)h$ pour $i = 1, \ldots, N+2$. En chaque point $t_i$, $i = 2, \ldots, N+1$, la valeur $u''(t_i)$ est approchée par le quotient $\dfrac{u(t_{i-1}) - 2u(t_i) + u(t_{i+1})}{h^2}$, (cf (10.4))

**1.** Montrer que le problème approché est donné par le système de $N$ équations à $N$ inconnues :

$$(P_h):\ AV = h^2(F - \sin V) - B$$

où on a noté

$$A = \begin{pmatrix} -2 & 1 & & 0 \\ 1 & \ddots & \ddots & \\ & \ddots & \ddots & 1 \\ 0 & & 1 & -2 \end{pmatrix} \in \mathbb{R}^{N,N},\ F = \begin{pmatrix} f(t_2) \\ \vdots \\ f(t_k) \\ \vdots \\ f(t_{N+1}) \end{pmatrix} \in \mathbb{R}^N,\ B = \begin{pmatrix} \alpha \\ 0 \\ \vdots \\ 0 \\ \beta \end{pmatrix} \in \mathbb{R}^N$$

$V = [v_2, ..., v_{N+1}]^T$ représente la valeur approchée de $u(t_2), \ldots, u(t_{N+1})$ à déterminer, $Vh = [\alpha; V; \beta]$ représente alors la solution approchée du problème (P).

L'équation $(P_h)$ n'est pas linéaire. On la résout par une méthode itérative qui consiste à choisir $V_0 = [0, \ldots, 0]^T$ puis à définir $V_{n+1}$ tel que :

$$AV_{n+1} = h^2(F - \sin V_n) - B \tag{10.6}$$

On admet que la méthode a convergé au bout d'un certain nombre d'itérations $n_0$ où $n_0$ est fixé à l'avance.

**2.** **Programmation**

Contruire le vecteur $t$. Tester avec $N = 5$, $a = 0$ et $b = 1$. Sauvegarder sous `pendul1.m`.

```
>> pendul1
t =
         0
    0.1667
    0.3333
    0.5000
    0.6667
    0.8333
    1.0000
```

**3.** On choisit $\alpha = 1$ et $\beta = -1$ et on définit $f(t) = \sin(\sin(\pi(t + 1/2))) - \pi^2 \sin(\pi(t + 1/2))$. La solution exacte est alors définie par $u(t) = \sin(\pi(t + 1/2))$.

Construire deux fonctions `f.m` et `u.m`. Test `f([0.1,1]), u([0.1,1])`.

**4.** En complétant le 1er programme, construire le vecteur $F$ de dimension $N$ et le vecteur $U = [u(t_1), \ldots, u(t_{N+2})]'$ de dimension $N + 2$. Tester avec $N = 5$, $a = 0$, $b = 1$. Afficher $F$ et $U$. Sauvegarder sous `pendul2.m`.

```
>> pendul2
F =
   -7.7856
   -4.4554
   -0.0000
    4.4554
    7.7856

U =
    1.0000
    0.8660
    0.5000
    0.0000
   -0.5000
   -0.8660
   -1.0000
```

**5.** Par ajout au 2e programme, construire la matrice $A$. Tester avec $N = 5$. Sauvegarder sous `pendul3.m`.

```
>> pendul3
A =
     -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
```

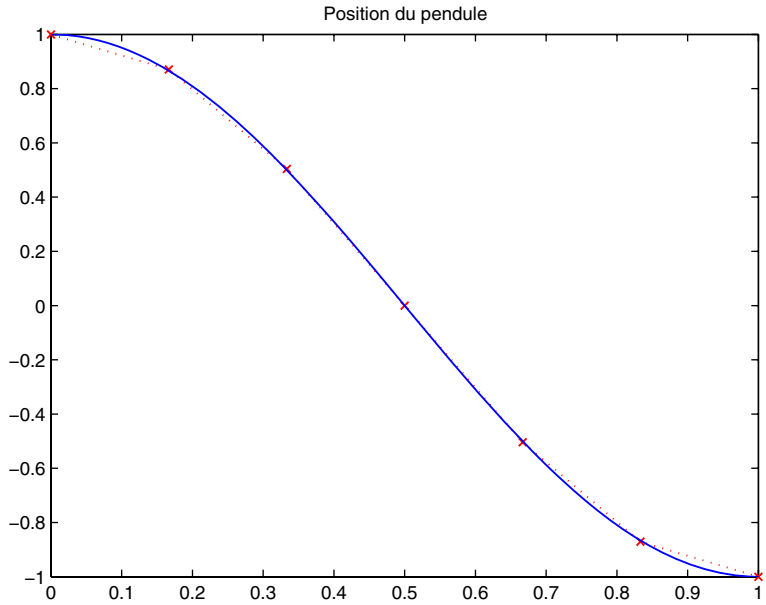
**6.** Résoudre l'équation (10.6) et programmer la méthode itérative.

Tester avec $n_0 = 6.$ et les valeurs précédentes. Afficher le vecteur $erriter$, calculer le vecteur $Vh = [\alpha; V_{n_0+1}; \beta]^T$ et l'afficher. Sauvegarder sous `pendul4.m`.

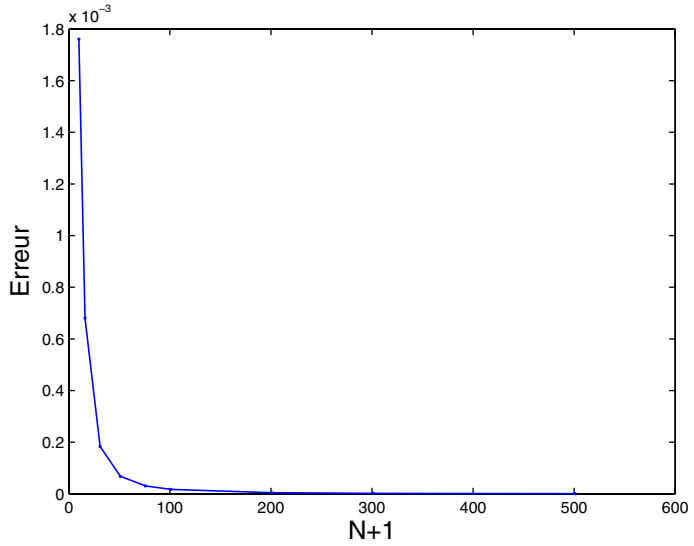
```
>> pendul4
erriter =
    0.8521    0.0183    0.0004    0.0000    0.0000    0.0000
```

**7.** Dessiner sur une même figure la solution exacte et la solution approchée. Calculer l'erreur $\|U - Vh\|_\infty = \max_{1 \leqslant i \leqslant N+2} |u(t_i) - Vh_i|$. Tester avec $N = 5, n_0 = 6$. Sauvegarder sous `pendul5.m`.

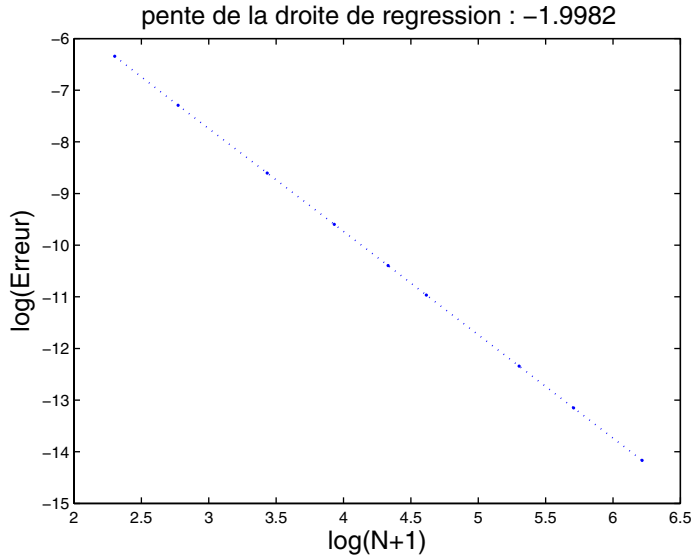
```
>> pendul5
erreur =
    0.0047
```




**8.** En faisant varier $N$, montrer graphiquement que $\|U - Vh\| \underset{N \to \infty}{\longrightarrow} 0$. Sauvegarder sous `pendul6.m`.




**9.** En rangeant les erreurs $\|U - Vh\|_\infty$ , pour différentes valeurs de $N$, dans un vecteur $taberr$, étudier la vitesse de convergence vers zéro de $\|U - Vh\|_\infty$ pour déterminer l'ordre de la méthode. On pourra prendre $n_0 = 2N$. Sauvegarder sous `pendul7.m`.




On trouve donc que l'erreur varie comme $h^2 = 1/(N+1)^2$.

**10.** L'équation du pendule amorti s'écrit :

$$(P') \begin{cases} u''(t) + u'(t) + \sin(u(t)) = f(t),\ t \in [a,b] \\ u(a) = \alpha,\ u(b) = \beta \end{cases}$$

On prendra : $a = 0, b = 1, \alpha = 0, \beta = 0$
et $f(t) = \sin(\cos(\pi(t + 1/2))) - \pi \sin(\pi(t + 1/2)) - \pi^2 \cos(\pi(t + 1/2))$.
Ainsi la solution exacte est : $u(t) = \cos(\pi(t + 1/2))$.

Pour $i = 2, ..., N + 1$ , on approche la valeur $u'(t_i)$ par $\dfrac{u(t_{i+1}) - u(t_{i-1})}{2h}$, ce qui conduit à résoudre le système $N \times N$ :

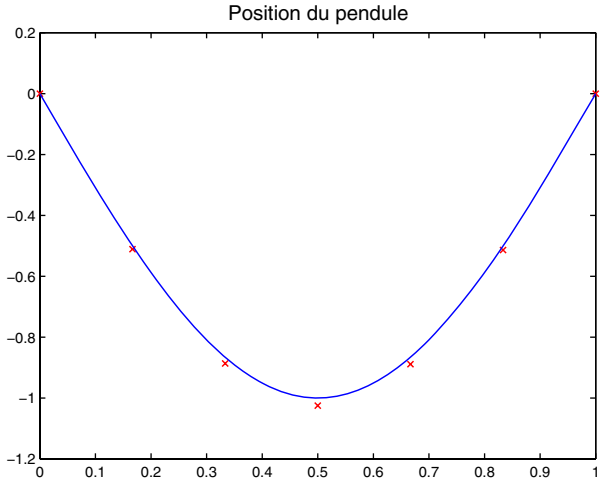
$$(P'_h) : (A + \frac{h}{2}C)V = h^2(F - \sin V) - \frac{h}{2}D - B$$

où :

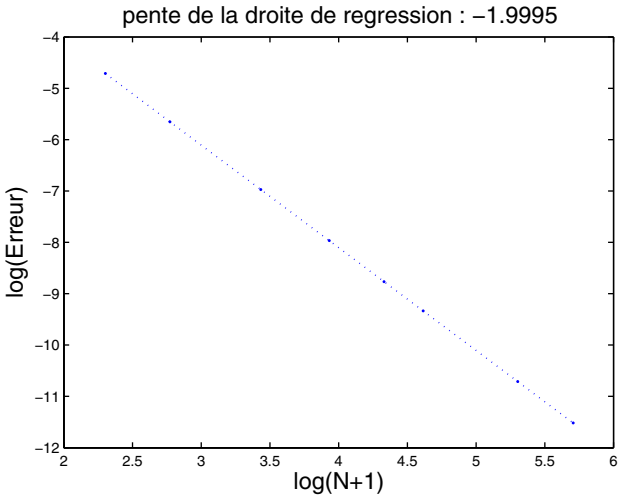
$$C = \begin{pmatrix} 0 & 1 & & 0 \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & 1 \\ 0 & & -1 & 0 \end{pmatrix} \in \mathbb{R}^{N\times N} \text{ et } D = \begin{pmatrix} -\alpha \\ 0 \\ \vdots \\ 0 \\ \beta \end{pmatrix} \in \mathbb{R}^N$$

Reprendre la question précédente pour calculer la solution du problème $(P'_h)$. En particulier, on affichera sur un même graphique la solution exacte et la solution approchée pour différentes valeurs de $N$. On étudiera aussi la vitesse de convergence vers zéro de $\|U_{ex} - U_h\|_\infty$. Sauvegarder sous `pendul8.m`.

```
>> pendul91
erriter =
    0.9346    0.0846    0.0056    0.0004    0.0000    0.0000

erreur =
    0.0252
```



`pendul92.m`, Étude d'erreur :





On retrouve une erreur d'ordre 2.

## 10.3 Schéma de Numerov

On se propose de déterminer une solution approchée du problème :

$$(P): \begin{cases} -u''(x) + c(x)u(x) = f(x),\ x \in [0,1] \\ u(0) = \alpha,\ u(1) = \beta. \end{cases}$$

où $c$ et $f$ sont 2 fonctions régulières avec $c \geqslant 0$. $(P)$ admet une solution unique notée $u$.
Soit $N$ un entier. On discrétise $[0,1]$ un pas de $h = \frac{1}{N+1}$ et on note $x = (x_1, ..., x_{N+2})^T$ où $x_i = (i-1)h$, $i = 1, \ldots, N+2$.

**1.** En chaque point $x_i$, $i = 2, \ldots, N+1$, pour une fonction $\phi$ de classe $\mathcal{C}^6$, montrer que

$$\left| \frac{1}{h^2}(\phi(x_{i-1}) - 2\phi(x_i) + \phi(x_{i+1})) - \frac{1}{12}(\phi''(x_{i-1}) + 10\phi''(x_i) + \phi''(x_{i+1})) \right| \leqslant \gamma h^4 \sup_{x \in [x_{i-1}, x_{i+1}]} \left| \phi^{(6)}(x) \right|$$

**2.** On notera $V = (v_1, ..., v_{N+2})^T$ la solution approchée de $U = (u(x_1), ..., u(x_{N+2}))^T$. Naturellement $v_1 = \alpha$ et $v_{N+2} = \beta$. Il reste à déterminer $W = (v_2, ..., v_{N+1})^T \in \mathbb{R}^N$. Montrer qu'en utilisant la formule précédente le problème approché est

$$(P_h):\ AW = b$$

où

$$A = \begin{pmatrix} 2 & -1 & & 0 \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & -1 \\ 0 & & -1 & 2 \end{pmatrix} + \frac{1}{12}\, h^2 \begin{pmatrix} 10c(x_2) & c(x_3) & & 0 \\ c(x_2) & 10c(x_3) & \ddots & \\ \ddots & \ddots & & c(x_{N+1}) \\ 0 & c(x_N) & & 10c(x_{N+1}) \end{pmatrix} \in \mathbb{R}^{N,N}$$

et

$$b = \frac{1}{12}\, h^2 \begin{pmatrix} f(x_1) + 10f(x_2) + f(x_3) \\ \vdots \\ \vdots \\ f(x_N) + 10f(x_{N+1}) - f(x_{N+2}) \end{pmatrix} + \begin{pmatrix} (1 - \frac{h^2}{12}\, c(x_1))\alpha \\ 0 \\ \vdots \\ 0 \\ (1 - \frac{h^2}{12}\, c(x_{N+2}))\beta \end{pmatrix} \in \mathbb{R}^N$$

Finalement le solution approchée est $V = \begin{pmatrix} \alpha \\ W \\ \beta \end{pmatrix}$.

### 3. Programmation

Pour $N$ donné, construire $X$, $F = (f(x_1), \ldots, f(x_{N+2}))^T$ et $C = (c(x_1) \ldots c(x_{N+2}))^T$. On construira aussi 2 fichiers `f.m` et `c.m`. Attention $X$, $F$ et $C$ sont dans $\mathbb{R}^{N+2}$. Sauvegarder sous `numerov1.m`. Test : $N = 5$, $f(x) = (\pi^2 + x)\sin(\pi(x + 1/2))$, $c(x) = x$.

```
>> numerov1
N =
     5

C =
         0
    0.1667
    0.3333
    0.5000
    0.6667
    0.8333
    1.0000

F =
    9.8696
    8.6917
    5.1015
    0.0000
   -5.2681
   -9.2690
  -10.8696
```

**4.** Compléter le programme précédent en construisant les 2 matrices de $R^{N\times N}$

$$A_1 = \begin{pmatrix} 2 & -1 & & & 0 \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & -1 \\ 0 & & & -1 & 2 \end{pmatrix}, \; A_2 = \begin{pmatrix} 10c(x_2) & c(x_3) & & 0 \\ c(x_2) & 10c(x_3) & \ddots & \\ \ddots & \ddots & & c(x_{N+1}) \\ 0 & c(x_N) & & 10c(x_{N+1}) \end{pmatrix}$$

et $A = A_1 + \dfrac{h^2}{12} A_2$ que l'on affichera. Sauvegarder sous `numerov2.m` et test avec $N = 5$

```
>> numerov2
A =

    2.0039   -0.9992         0         0         0
   -0.9996    2.0077   -0.9988         0         0
         0   -0.9992    2.0116   -0.9985         0
         0         0   -0.9988    2.0154   -0.9981
         0         0         0   -0.9985    2.0193
```

**5.** Compléter le programme ci-dessus avec le calcul de

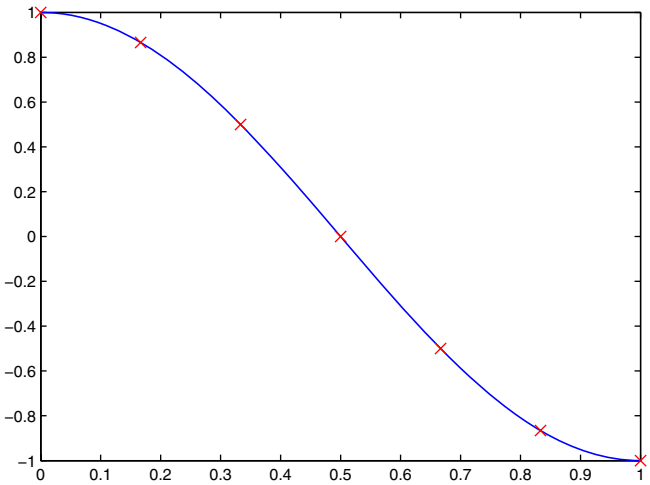
$$b_1 = \begin{pmatrix} f(x_1) + 10f(x_2) + f(x_3) \\ \vdots \\ \vdots \\ f(x_N) + 10f(x_{N+1}) + f(x_{N+2}) \end{pmatrix} \in \mathbb{R}^N, \; b_2 = \begin{pmatrix} (1 - \frac{h^2}{12} c(x_1))\alpha \\ 0 \\ \vdots \\ 0 \\ (1 - \frac{h^2}{12} c(x_{N+2}))\beta \end{pmatrix} \in \mathbb{R}^N$$

et $b = \dfrac{h^2}{12} b_1 + b_2$ que l'on affichera. Sauvegarder sous `numerov3.m` et test avec $N = 5$, $\alpha = 1$, $\beta = -1$.

```
>> numerov3
b =
    1.2359
    0.1382
   -0.0004
   -0.1434
   -1.2496
```

**6.** Déterminer et afficher $W$, $V$ et $err = \|V - U\|_\infty$. Dessiner solutions exacte et approchée. Sauvegarder sous `numerov4.m` et test avec $N = 5$. La solution exacte est donnée par $u(x) = \sin(\pi(x + 1/2))$.

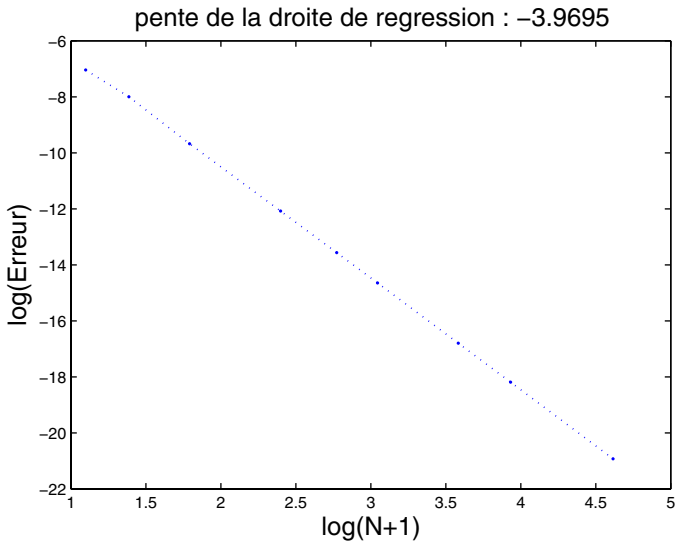
```
>> numerov4
V =

    1.0000
    0.8661
    0.5001
    0.0000
   -0.5001
   -0.8661
   -1.0000

err =
  6.3119e-005
```




**7.** Inventer un exemple $f_1$, $u_1$, $c_1$ où $\|V - U\|_\infty = 0$ sauf erreurs machines.

**8.** Étudier l'erreur de méthode $\|V - U\|_\infty$ quand $N$ varie. En particulier déterminer $k$ tel que $err \simeq \dfrac{\lambda}{(N+1)^k}$. Sauvegarder sous `numerov5.m`. Test avec les données des questions 1 à 4 sauf $tN = [2, 3, 5, 10, 15, 20, 35, 50, 100]$.

```
taberr =
  1.0e-003 *
    0.8749    0.3368    0.0631    0.0057    0.0013    0.0004...
        0.0001    0.0000    0.0000
```




On trouve que log(*err*) est sensiblement de la forme $-4\log(N+1) + cste$, soit $err \simeq \dfrac{\lambda}{(N+1)^4}$. Apparemment, la méthode est d'ordre 4. On peut reprendre une étude de l'erreur semblable à celle de la poutre pour le démontrer.

**9.** On considère maintenant le problème

$$(P') : \begin{cases} \dfrac{\partial u}{\partial t} - \dfrac{\partial^2 u}{\partial x^2} + c(x)u = 0,\ x \in [0,1],\ 0 \leqslant t \leqslant T \\ u(t,0) = \alpha \\ u(t,T) = \beta \\ u(0,x) = \alpha + (\beta - \alpha)x \end{cases}$$

On approche $\dfrac{\partial u}{\partial t}(t,x)$ par $\dfrac{u(t,x) - u(t-\tau,x)}{\tau}$. On peut alors construire un schéma implicite en temps

$$(A + \frac{h^2}{\tau} B)W_{p+1} = \frac{h^2}{\tau} B W_p + b_2$$

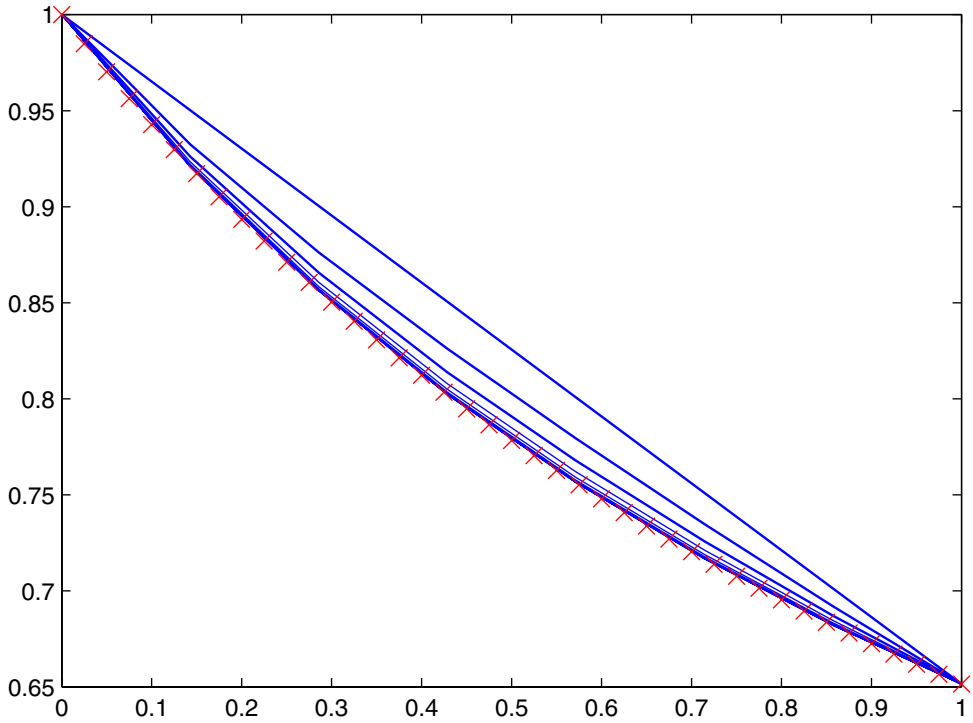
où $B = \dfrac{1}{12} \begin{pmatrix} 10 & 1 & & 0 \\ 1 & \ddots & \ddots & \\ & \ddots & & 1 \\ 0 & & 1 & 10 \end{pmatrix} \in \mathbb{R}^{N,N}$

et où $W_{p+1}$ est une approximation de $(u((p+1)\tau, x_2) \ldots u((p+1)\tau, x_{N+1}))^T$ .

Écrire dans `Numerov6.m` un programme calculant une approximation de $u(T,x)$ pour un pas de temps $\tau$ donné. Test avec $N = 6$, $\alpha = 1$, $\tau = 1/10$, $T = 1$ , $\beta = 2^{(1-\sqrt{5})/2}$ et $c1(x) = \dfrac{1}{(x+1)^2}$.

**10.** La solution stationnaire est $u_\infty(x) = (x+1)^{\frac{1-\sqrt{5}}{2}}$. Montrer, sur un graphique avec plusieurs courbes, que $u(t,x)$ tend vers $u_\infty(x)$

`N=6, tau=1/10, T=1`, solutions approchées en $0, 1/10, \ldots, 1$ et solution stationnaire (++).




## 10.4 Équation de convection-diffusion

On souhaite résoudre le problème suivant, équation de convection-diffusion :

$$(P)\begin{cases} -u''(t) + rc(t)u'(t) = & f(t), \quad t \in [0,1] \\ u(0) = & 0 \\ u(1) = & 0 \end{cases}$$

On construit un problème approché par une méthode de différences finies. Pour cela, l'intervalle $[0,1]$ est discrétisé avec un pas constant $h = 1/(N+1)$. On appelle $T = (t_1, \ldots, t_{N+2})^T$ le vecteur tel que $t_i = (i-1)h,\ i = 1, \ldots, N+2$. En chaque point $t_i,\ i = 2, \ldots, N+1$, la valeur $u''(t_i)$ est approchée par le quotient $\dfrac{u(t_{i-1}) - 2u(t_i) + u(t_{i+1})}{h^2}$ et la dérivée $u'(t_i)$ est approchée par $\dfrac{u(t_{i+1}) - u(t_{i-1})}{2h}$

**1.** Montrer que le problème approché est donné par le système de $N$ équations à $N$ inconnues :

$$(P_h) : \ MV = F \tag{10.7}$$

où $M = A + \dfrac{rh}{2} B$ avec

$$A = \begin{pmatrix} 2 & -1 & 0 & \dots & 0 \\ -1 & 2 & -1 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & -1 \\ 0 & \dots & 0 & -1 & 2 \end{pmatrix} \in \mathbb{R}^{N,N}, \ F = h^2 \begin{pmatrix} f(t_2) \\ \vdots \\ f(t_k) \\ \vdots \\ f(t_{N+1}) \end{pmatrix} \in \mathbb{R}^N$$

$$B = \begin{pmatrix} c(t_2) & 0 & 0 & \dots & 0 \\ 0 & c(t_3) & 0 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & 0 & c(t_{N+1}) \end{pmatrix} \times \begin{pmatrix} 0 & 1 & 0 & \dots & 0 \\ -1 & 0 & 1 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & 0 & 1 \\ 0 & \dots & 0 & -1 & 0 \end{pmatrix} \in \mathbb{R}^{N,N},$$

$V = (v_2, ..., v_{N+1}))^T$ représente la valeur approchée de $(u(t_2), \dots, u(t_{N+1}))^T$ à déterminer.
La solution approchée du problème est alors $Vh = [0; V; 0]$.

**2.** **Programmation**
Contruire le vecteur $T \in \mathbb{R}^{N+2}$. Tester avec $N = 4$. Sauvegarder sous `CD1.m`.

```
>> CD1
T =
                0
                0.2000
                0.4000
                0.6000
                0.8000
                1.0000
```

**3.** On définit $f(r,t) = r^2\, e^{r\,t}\,(t-1)/(1-e^r) + rt$ et $c(t) = t$. La solution exacte est définie par : $u(r,t) = t - (1-e^{r\,t})/(1-e^r)$. *On a remplacé $f(t)$ par $f(r,t)$ et de même pour $u$.*

Construire trois fonctions `f.m` calculant $f(r,T)$, `u.m` calculant $u(r,T)$ et `c.m` calculant $c(T)$. Tester avec : $r = 1$, $T = 0:3$.

```
f1 =     0.5820    1.0000   -2.3003  -20.3786
c1 =     0     1     2     3
u1 =     0     0    -1.7183   -8.1073
```

**4.** En complétant `CD1.m`, construire le vecteur $F$ et le vecteur $C = [c(t_2), \ldots, c(t_{N+1})]^T$ de dimension $N$ et la valeur $Uex = [u(r, t_1), \ldots, u(r, t_{N+2})]^T$ de dimension $N+2$. Tester avec $r = 1$ et $N = 4$. Afficher $F$, $C$ et $Uex$. Sauvegarder sous `CD3.m`

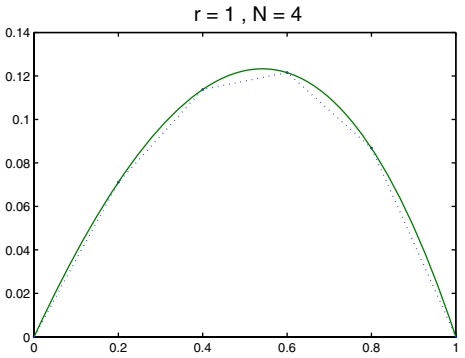
```
>> CD3
F =
    0.0307
    0.0368
    0.0410
    0.0424
C =
    0.2000
    0.4000
    0.6000
    0.8000
Uex =
         0
    0.0711
    0.1138
    0.1215
    0.0868
         0
```

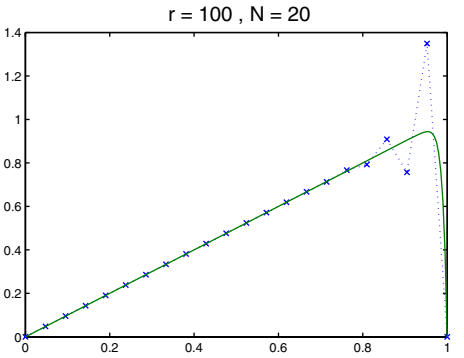
**5.** Par ajout au programme précédent, construire les matrices $A$ et $B$ puis $M$. Afficher $M$. *Indication* : utiliser diag, ones. Tester avec $r = 1$ et $N = 4$. Sauvegarder sous `CD4.m`.

```
>> CD4
M =
    2.0000   -0.9800         0         0
   -1.0400    2.0000   -0.9600         0
         0   -1.0600    2.0000   -0.9400
         0         0   -1.0800    2.0000
```

**6.** Résoudre l'équation (10.7) et construire la solution approchée $V_h$ dans $\mathbb{R}^{N+2}$. Dessiner la solution approchée et la solution exacte. Tester avec $r = 1$ et $N = 5$. Afficher $V_h$ et le graphe. Sauvegarder sous `CD5a.m`. Tester aussi avec $r = 100$ et $N = 20$. Afficher le graphe. Sauvegarder sous `CD5b.m`.

```
>> CD5
N =
     4
r =
     1
Vh =
         0
    0.0711
    0.1138
    0.1216
    0.0868
         0
```

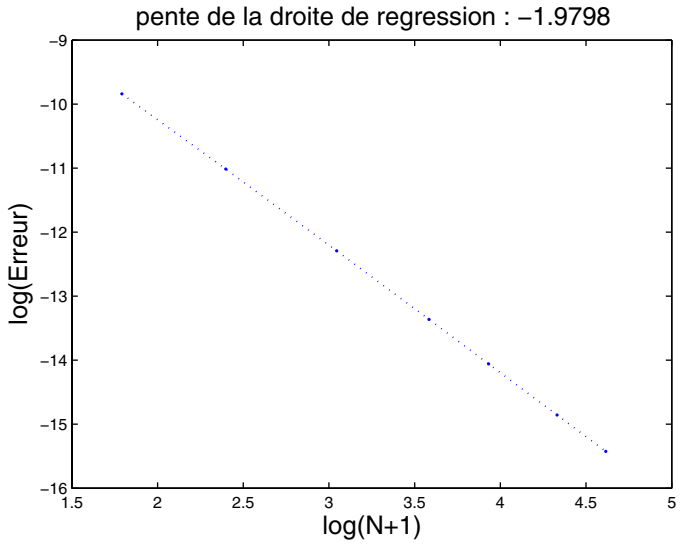



**7.** Ajouter le calcul d'erreur $\|V_h - Uex\|_\infty$. Tester avec $r = 1$ et $N = 4$. Afficher l'erreur. Sauvegarder sous `CD6.m`.

```
>> CD6
err =
   7.8507e-005
```

**8.** Avec $r = 1$, à partir du tableau $tN = [5,\ 10,\ 20,\ 35,\ 50,\ 75,\ 100]$, construire un tableau $taberr$ puis tracer le graphe de $\log(taberr)$ en fonction de $\log(tN + 1)$. Sauvegarder sous `CD7.m`. Conclure sur l'ordre de la méthode.



La méthode est d'ordre 2.

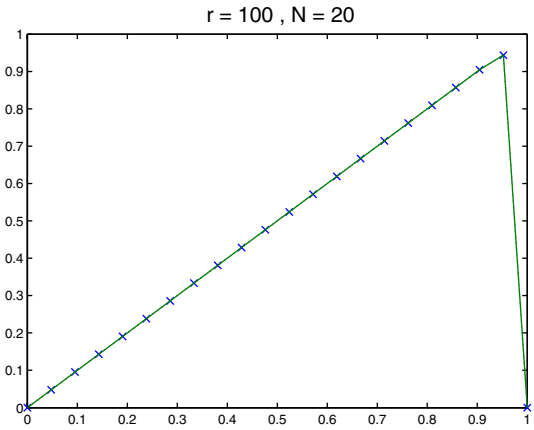
**9.** Pour éviter le phénomène découvert à la question 5, on modifie le problème approché en :

$$(Q_h) : \ M_1 V = F \tag{10.8}$$

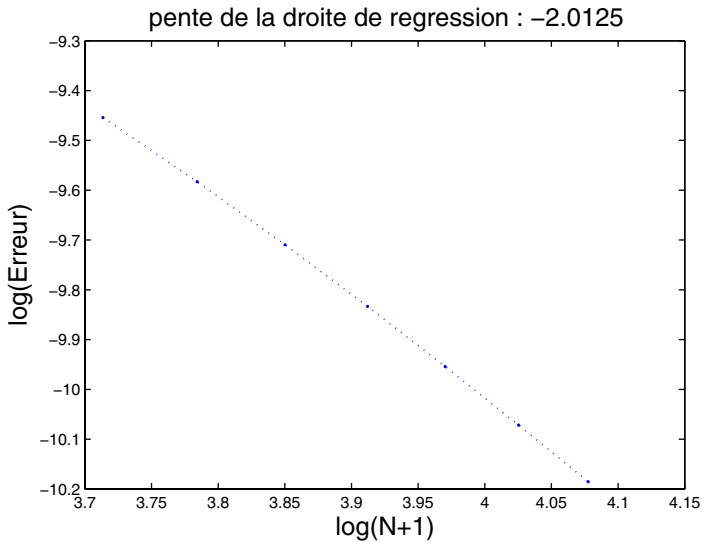
où $M_1 = A + rhB_1$ en définissant $B_1$ par

$$\begin{pmatrix} c(t_2) & 0 & 0 & \dots & 0 \\ 0 & c(t_3) & 0 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & 0 & c(t_{N+1}) \end{pmatrix} \times \begin{pmatrix} 2a_1 - 1 & 1 - a_1 & 0 & \dots & 0 \\ -a_2 & 2a_2 - 1 & 1 - a_2 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & -a_{N-1} & 2a_{N-1} - 1 & 1 - a_{N-1} \\ 0 & \dots & 0 & -a_N & 2a_N - 1 \end{pmatrix},$$

où $G = rh[c(t_2), \dots, c(t_{N+1})]^T = rhC \in \mathbb{R}^N$ puis $a = 1/2 + 1/2 \coth(G/2) - 1./G \in \mathbb{R}^N$. Construire la matrice $M_1$. Déterminer la solution approchée. Tracer le graphe. Afficher $V_h$. Tester avec $r = 100$ et $N = 20$. Afficher le graphe. Sauvegarder sous `CD8.m`. *À noter que plus généralement, si* $c(t_i) = 0$ *alors* $a_{i-1}$ *correspondant n'est pas défini ; on prend* $a_{i-1} = 1/2$ *dans ce cas.*




**10.** Étudier l'erreur pour $tN = [40 : 3 : 60]$. Sauvegarder sous `CD9.m` . Conclure.




## DU MAL À DÉMARRER ?

### 10.1 Flexion d'une poutre

**1.** Commencer par $-u'' + cu = 0$ en prenant $c = \omega^2$, puis chercher une solution particulière de $-u'' + cu = f$ puis toutes les solutions.

**2.** On pourra commencer par un développement de Taylor de $\phi(x+h)$ puis $\phi(x-h)$ et pour conclure, ne pas oublier que $\phi^{(4)}$ est continue.

**4.** $$A_h = \begin{pmatrix} 2+ch^2 & -1 & 0 & \dots & 0 \\ -1 & 2+ch^2 & -1 & 0 & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & 0 & -1 & 2+ch^2 & -1 \\ 0 & \dots & 0 & -1 & 2+ch^2 \end{pmatrix}.$$

**5.** (a) Étudier la ligne $i$ de $A_h V$ où $v_i = \min_{j=1,\dots,n} v_j$ pour montrer que $v_i \geqslant 0$.

### 10.2 Oscillations d'un pendule

**1.** $\dfrac{u(t_{i-1}) - 2u(t_i) + u(t_{i+1})}{h^2} - \dfrac{h^2}{12} u^{(4)}(t_i + \theta_i h) + sin(u(t_i)) = f(t_i)$. Approcher $u(t_j)$ par $v_j$ en négligeant $\dfrac{h^2}{12} u^{(4)}(t_i + \theta_i h)$.

**3.** Utiliser `diag, ones`.

**6.** À chaque itération, déterminer $X_1$ tel que $AX_1 = h^2(F - \sin X_0) - B$ où $X_0$ est l'ancienne valeur i.e. $V_n$ et $X_1$ est la nouvelle valeur i.e. $V_{n+1}$, évaluer l'erreur en norme infinie $\|X_1 - X_0\|_\infty$, entre l'ancienne et la nouvelle valeur ; les erreurs seront rangées dans un vecteur *erriter* à $n_0$ composantes. Remettre à jour : $X_0 = X_1$.

### 10.3 Schéma de Numerov

**1.** On pourra commencer par des développements de Taylor à l'ordre 6 de $\phi(x_{i-1})$, $\phi(x_{i+1})$, $\phi''(x_{i-1})$ et $\phi''(x_{i+1})$.

**2.** Reprendre les idées de l'exercice 10.2.

### 10.4 Équation de convection-diffusion

**4.** Reprendre encore les idées de l'exercice 10.2.

## CORRIGÉS DES EXERCICES

### 10.1 Flexion d'une poutre

**1.** Sachant que $c > 0$, nous pouvons écrire $c = \omega^2$, avec $\omega > 0$. L'ensemble des solutions de $-u'' + cu = 0$ est un espace vectoriel de dimension 2 dont les éléments sont de la forme $u(x) = \lambda e^{\omega x} + \mu e^{-\omega x}$. Si nous cherchons une solution particulière de $-u'' + cu = f$ sous la forme $u(x) = \lambda(x)e^{\omega x}$, nous obtenons $\lambda''(x) + 2\omega\lambda'(x) = -e^{-\omega x} f(x)$, équation différentielle linéaire du premier ordre en $\lambda'$. Les solutions de $\lambda''(x) + 2\omega\lambda'(x) = 0$ sont alors de la forme $\lambda'(x) = e^{-2\omega x}\mu$. En cherchant une solution particulière sous la forme $\lambda'(x) = e^{-2\omega x}\mu(x)$, il vient $\mu'(x) = -e^{\omega x} f(x)$ et par exemple $\mu(x) = -\displaystyle\int_0^x e^{\omega s} f(s)ds$ puisque $f$ est continue. On peut choisir $\lambda(x) = -\displaystyle\int_0^x \left( e^{-2\omega t} \int_0^t e^{\omega s} f(s)ds \right) dt$ et finalement la solution générale de $-u'' + cu = f$ est de la forme

$$u(x) = \lambda e^{\omega x} + \mu e^{-\omega x} - e^{\omega x} \int_0^x \left( e^{-2\omega t} \int_0^t e^{\omega s} f(s)ds \right) dt.$$

Les conditions aux bords $u(0) = u(1) = 0$ donneront un système de 2 équations linéaires à 2 inconnues $\lambda$ et $\mu$ :

$$\begin{cases} \lambda + \mu = 0 \\ \lambda e^{\omega} + \mu e^{-\omega} = e^{\omega} \displaystyle\int_0^1 \left( e^{-2\omega t} \int_0^t e^{\omega u} f(u)du \right) dt \end{cases}$$

dont le déterminant est non nul et qui admet alors une solution unique. Nous pouvons conclure que $(P)$ admet une solution unique.

**2.** Soit $\phi$ une fonction de classe $\mathcal{C}^4$ au voisinage d'un point $x$ réel. Pour $h$ suffisamment petit, $\phi \in \mathcal{C}^4[x-h, x+h]$. En utilisant la formule de Taylor, il existe $\theta_1$ et $\theta_2$ dans $]0,1[$ tels que

$$\begin{aligned}\phi(x+h) &= \phi(x) + h\phi'(x) + \frac{h^2}{2}\phi''(x) + \frac{h^3}{6}\phi^{(3)}(x) + \frac{h^4}{24}\phi^{(4)}(x+\theta_1 h)\\ \phi(x-h) &= \phi(x) - h\phi'(x) + \frac{h^2}{2}\phi''(x) - \frac{h^3}{6}\phi^{(3)}(x) + \frac{h^4}{24}\phi^{(4)}(x-\theta_2 h)\end{aligned}$$

En ajoutant ces 2 égalités et en soustrayant $2\phi(x)$, puis en divisant par $h^2$, on obtient $\phi''(x) = \dfrac{\phi(x-h) - 2\phi(x) + \phi(x+h)}{h^2} - \dfrac{h^2}{24}\left(\phi^{(4)}(x+\theta_1 h) + \phi^{(4)}(x-\theta_2 h)\right)$. Puisque $\phi^{(4)}$ est continue, le théorème des valeurs intermédiaires nous assure qu'il existe un point entre $x-\theta_2 h$ et $x+\theta_1 h$ que nous notons $x+\alpha h$ et qui vérifie $\phi^{(4)}(x+\alpha h) = \dfrac{\phi^{(4)}(x+\theta_1 h) + \phi^{(4)}(x-\theta_2 h)}{2}$ ce qui permet de conclure que

$$\phi''(x) = \frac{\phi(x-h) - 2\phi(x) + \phi(x+h)}{h^2} - \frac{h^2}{12}\phi^{(4)}(x+\alpha h),\ \alpha \in ]-\theta_2, \theta_1[ \subset ]-1, 1[.$$

**3.** Sachant que $-u''(x) + cu(x) = f(x)$, pour $i = 1, \ldots, N$ en $x_i$, on obtient

$$\begin{aligned}&-\frac{u(x_{i-1}) - 2u(x_i) + u(x_{i+1})}{h^2} + \frac{h^2}{12}u^{(4)}(x_i + \alpha_i h) + cu(x_i) = f(x_i)\\ \Leftrightarrow\quad &-u(x_{i-1}) + (2 + h^2 c)u(x_i) - u(x_{i+1}) + \frac{h^4}{12}u^{(4)}(x_i + \alpha_i h) = h^2 f(x_i)\end{aligned}$$

À noter que pour $i = 1$ la valeur $u(x_0)$ est connue, ici 0 et de même pour $i = N$ avec $u(x_{N+1}) = 0$ ; ces termes peuvent donc disparaître de l'écriture.

**4.** Si on imagine que $v_i$ est une approximation de $u(x_i)$ incluant l'erreur faite sur la dérivée 4ième, on obtient $-v_{i-1} + (2 + h^2 c)v_i - v_{i+1}h^2 = h^2 f(x_i)$ pour $i = 1, \ldots, N$. Ces $N$ équations s'écrivent $A_h V_h = F_h$ où $V_h = (v_1, \ldots, v_N)^T$ est l'approximation de $U_h = (u(x_1), \ldots, u(x_N))^T$ cherchée,

$$A_h = \begin{pmatrix} 2+ch^2 & -1 & 0 & \ldots & 0 \\ -1 & 2+ch^2 & -1 & 0 & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & 0 & -1 & 2+ch^2 & -1 \\ 0 & \ldots & 0 & -1 & 2+ch^2 \end{pmatrix} \in \mathbb{R}^{N\times N},\ F_h = \begin{pmatrix} f(x_1) \\ f(x_2) \\ \vdots \\ \vdots \\ f(x_N) \end{pmatrix} \in \mathbb{R}^N.$$

La matrice $A_h$ est symétrique, tridiagonale et à diagonale dominante stricte i.e. pour tout $i$, $a_{ii} > \displaystyle\sum_{j=1,\ j\neq i}^{n} |a_{ij}|$ car $c > 0$. Nous avons déjà rencontré une matrice à diagonale dominante

stricte dans l'étude des splines cubiques. Ces matrices sont inversibles, mais nous allons utiliser une autre méthode pour montrer que $A_h$ est inversible et que donc le système admet une solution unique.

**5.** Supposons que $A_h V \geqslant 0$ et appelons $i$ un indice tel que $v_i = \min_{j=1,\dots,n} v_j$. Tout d'abord, nous supposons que $i \neq 1$ et $i \neq n$. Nous avons $-v_{i-1} + (2 + h^2 c)v_i - v_{i+1} \geqslant 0$ soit encore $v_i - v_{i-1} + h^2 c v_i + v_i - v_{i+1} \geqslant 0$. Or $v_i - v_{i-1} \leqslant 0$ et $v_i - v_{i+1} \leqslant 0$ puisque $i$ réalise le minimum. Donc nécessairement $h^2 c v_i \geqslant 0$. Or $h^2 c > 0$ d'où $v_i \geqslant 0$. On peut faire un raisonnement similaire si $i = 1$ ou $i = n$. On en déduit que $v_i = \min_{j=1,\dots,n} v_j \geqslant 0$ et donc $V \geqslant 0$.

Si $A_h V = 0$ alors $A_h V \geqslant 0$ donc $V \geqslant 0$. Mais nous avons aussi $A_h(-V) = -A_h V = 0$ donc $-V \geqslant 0$. Le vecteur $V$ a ses composantes négatives et positives, il est donc nul. On peut conclure que $A_h$ est inversible. En écrivant $A_h^{-1} = (C_1 C_2 \dots C_N)$ en colonnes, puis en effectuant un produit colonne par colonne de $A_h(C_1 C_2 \dots C_N) = Id$, sachant que les colonnes de la matrices $Id$ sont positives, on en déduit que chacune des colonnes $C_j$ est positive donc $A_h^{-1} \geqslant 0$.

Enfin, nous avons vu que la matrice $A_h$ était inversible donc le système définissant $(P_h)$ admet une solution unique.

À noter que toutes les matrices à diagonale dominante stricte ne sont pas forcément monotones. Ainsi $A = \begin{pmatrix} 2 & 1 \\ 1 & 2 \end{pmatrix}$ vérifie $A \begin{pmatrix} -1 \\ 3 \end{pmatrix} = \begin{pmatrix} 1 \\ 5 \end{pmatrix} \geqslant 0$ et $A^{-1} = \frac{1}{3} \begin{pmatrix} 2 & -1 \\ -1 & 2 \end{pmatrix}$.

**6.** Soient $w$ et $\theta$ deux vecteurs tels que $A_h w = \theta$. Nous définissons la fonction polynômiale du second degré $\psi(x) = \dfrac{x(1-x)}{2h^2}\|\theta\|_\infty$ dont la dérivée seconde vaut $\psi'' = -\dfrac{\|\theta\|_\infty}{h^2}$. Notons que $\psi(x_0) = \psi(0) = 0$ et de même $\psi(x_{N+1}) = \psi(1) = 0$. Soit $\Psi$ le vecteur $(\psi(x_1), \dots, (\psi(x_N))^T$. Soit $\ell_i$ la ligne $i$ de $A_h \Psi$. $\ell_i = -\psi(x_{i-1}) + (2 + ch^2)\psi(x_i) - \psi(x_{i+1})$, soit, compte tenu de la question 2, et puisque $\psi^{(4)} = 0$, $\ell_i = -h^2\psi''(x_i) + ch^2\psi(x_i) = \|\theta\|_\infty + ch^2\psi(x_i) \geqslant \|\theta\|_\infty$ car $\psi$ est positive sur $[0, 1]$. On a donc $\ell_i \geqslant \theta_i$ et $\ell_i \geqslant -\theta_i$ puisque $\|\theta\|_\infty = \max_{i=1,\dots,N} |\theta_i|$. On en déduit $A_h \Psi \geqslant A_h w$ ou encore $A_h(\Psi - w) \geqslant 0$ qui permet d'obtenir $\psi \geqslant w$. De même par $A_h \Psi \geqslant A_h(-w)$, on obtient $\psi \geqslant -w$.

C'est dire que pour tout $i$, $\pm w_i \leqslant \psi(x_i) \leqslant \max_{x\in[0,1]} \psi(x) = \psi(1/2)$, d'où $\|w\|_\infty \leqslant \dfrac{\|\theta\|_\infty}{8h^2}$.

D'après la question 2, $A_h U_h = F_h - \dfrac{h^4}{12} \begin{pmatrix} u^{(4)}(x_1') \\ \dots \\ u^{(4)}(x_N') \end{pmatrix}$, donc $A_h(U_h - V_h) = -\dfrac{h^4}{12} \begin{pmatrix} u^{(4)}(x_1') \\ \dots \\ u^{(4)}(x_N') \end{pmatrix}$.

Finalement, en posant $w = U_h - V_h$ et $\theta = -\dfrac{h^4}{12} \begin{pmatrix} u^{(4)}(x_1') \\ \dots \\ u^{(4)}(x_N') \end{pmatrix}$, on déduit

$$\|U_h - V_h\|_\infty \leqslant \frac{1}{8h^2}\|\theta\|_\infty \leqslant \frac{1}{8h^2}\frac{h^4}{12} M_4 = h^2 \frac{M_4}{96}$$

où $M_4 = \max_{x\in[0,1]} |u^{(4)}(x)|$.

## 10.2 Oscillations d'un pendule

**1.** Pour $i = 2, \ldots, N+1$, en chaque $x_i$, nous avons $\dfrac{u(t_{i-1}) - 2u(t_i) + u(t_{i+1})}{h^2} - \dfrac{h^2}{12}u^{(4)}(t_i + \theta_i h) + sin(u(t_i)) = f(t_i)$. Si nous négligeons le terme $\dfrac{h^2}{12}u^{(4)}(t_i + \theta_i h)$, en notant $v_j$ l'approximation correspondante de $u(x_j)$, il vient :

$$\frac{v_{i-1} - 2v_i + v_{i+1}}{h^2} + sin(v_i) = f(t_i),\ i = 1, \ldots, N.$$

La première et la dernière équation sont un peu particulières. En $i = 2$, $v_{i-1} = v_1 = \alpha$ et de même en $i = N + 1$, $v_{N+2} = \beta$. Enfin, si on multiplie les équations par $h^2$, on peut alors les écrire sous la forme du système $AV = h^2(F - sin(V) + B$ où $A$, $F$ et $B$ sont définies précédemment.

**3.**

```
function v=f(t);
v=sin(sin(pi*(t+0.5)))-pi^2*sin(pi*(t+0.5));
```

```
function v=u(t);
v=sin(pi*(t+0.5));
```

**2.** **4.** **5.** **6.** **7.** `pendul5.m`

```
alpha=1;beta=-1;
a=0;b=1;
N=5;
n0=6;
h=(b-a)/(N+1);h2=h*h;
t=(a:h:b)'
t1=t(2:N+1);
F=f(t1)
U=u(t)
A=-2*diag(ones(1,N))+diag(ones(1,N-1),1)+diag(ones(1,N-1),-1)
Y0=zeros(N,1);
B=zeros(N,1);
B(1)=alpha;B(N)=beta;
for j=1:n0
    Z=h2*(F-sin(Y0))-B;
    Y1=A\Z;
    erriter(j)=norm(Y1-Y0,inf);
    Y0=Y1;
end;
erriter
Vh=[alpha;Y1;beta];
erreur=norm(U-Vh,inf)
tt=a:(b-a)/500:b;
plot(t,Vh,'rx',tt,u(tt))
title('Position du pendule')
```

### Ce qu'il faut retenir de cet exercice

La seule boucle utilisée est celle de la méthode itérative.

**8.** **9.** pendul7.m

```
clear
alpha=1;beta=-1;
a=0;b=1;
tN=[9,15,30,50,75,100,200,300,500];
for i=1:length(tN)
    N=tN(i);n0=2*N;
    h=(b-a)/(N+1);h2=h*h;
    t=(a:h:b)';
    t1=t(2:N+1);
    F=f(t1);
    A=-2*diag(ones(1,N))+diag(ones(1,N-1),1)+diag(ones(1,N-1),-1);
    Y0=zeros(N,1);
    B=zeros(N,1);
    B(1)=alpha;B(N)=beta;
    for j=1:n0
        Z=h2*(F-sin(Y0))-B;
        Y1=A\Z;
        Y0=Y1;
    end;
    Vh=[alpha;Y1;beta];
    U=u(t);
    terr(i)=norm(U-Vh,inf);
end;
tN,terr
%question 7
%plot(tN+1,terr)
plot(log(tN+1),log(terr))
a=polyfit(log(tN+1),log(terr),1)
title(['pente␣de␣la␣droite␣de␣regression␣:␣'...
    ,num2str(a(1))],'FontSize',16)
xlabel('log(N+1)','FontSize',16)
ylabel('log(Erreur)','FontSize',16)
```

**10.** Pour pendul91.m, dans pendul5.m, remplacer les lignes

```
A=-2*diag(ones(1,N))+diag(ones(1,N-1),1)+diag(ones(1,N-1),-1)
B=zeros(N,1);
B(1)=alpha;B(N)=beta;
```

par les lignes

```
A=-2*diag(ones(1,N))+diag(ones(1,N-1),1)+diag(ones(1,N-1),-1);
C=diag(ones(1,N-1),1)-diag(ones(1,N-1),-1);
Amod=A+h/2*C;
B=zeros(N,1);
B(1)=alpha;B(N)=beta;
```

```
D=zeros(N,1);
D(1)=-alpha;D(N)=beta;
Bmod=B+h/2*D;
```

et modifier les conditions initiales et les noms de variables dans la boucle.

**Ce qu'il faut retenir de cet exercice**

Si le programme initial `pendul5.m` a été bien fait, il n'y a pas grand chose à modifier.

Pour `pendul92.m`, mêmes changements à partir de `pendul7.m`

## 10.3 Schéma de Numerov

**1.** En chaque point $x_i$, $i = 2, \dots, N+1$, pour une fonction $\phi$ de classe $\mathcal{C}^6$, par la formule de Taylor, il existe $\alpha_j \in ]0,1[$, $j = 1,2,3,4$ tels que

$$\begin{aligned}
\phi(x_{i-1}) &= \phi(x_i) - h\phi'(x_i) + \frac{h^2}{2}\phi''(x_i) - \frac{h^3}{6}\phi^{(3)}(x_i) + \frac{h^4}{24}\phi^{(4)}(x_i) - \frac{h^5}{120}\phi(x_i) \\
&+ \frac{h^6}{720}\phi^{(6)}(x_i - \alpha_1 h), \\
\phi(x_{i+1}) &= \phi(x_i) + h\phi'(x_i) + \frac{h^2}{2}\phi''(x_i) + \frac{h^3}{6}\phi^{(3)}(x_i) + \frac{h^4}{24}\phi^{(4)}(x_i) + \frac{h^5}{120}\phi(x_i) \\
&+ \frac{h^6}{720}\phi^{(6)}(x_i + \alpha_2 h), \\
\phi''(x_{i-1}) &= \phi''(x_i) - h\phi^{(3)}(x_i) + \frac{h^2}{2}\phi^{(4)}(x_i) - \frac{h^3}{6}\phi^{(5)}(x_i) + \frac{h^4}{24}\phi^{(6)}(x_i - \alpha_3 h), \\
\phi''(x_{i+1}) &= \phi''(x_i) + h\phi^{(3)}(x_i) + \frac{h^2}{2}\phi^{(4)}(x_i) + \frac{h^3}{6}\phi^{(5)}(x_i) + \frac{h^4}{24}\phi^{(6)}(x_i + \alpha_4 h).
\end{aligned}$$

On en déduit, en utilisant le théorème des valeurs intermédiaires pour $\phi^{(6)}$ que

$$\begin{aligned}
&\frac{1}{h^2}\left(\phi(x_{i-1}) - 2\phi(x_i) + \phi(x_{i+1})\right) - \frac{1}{12}\left(\phi''(x_{i-1}) + 10\phi''(x_i) + \phi''(x_{i+1})\right) \\
= \;& \phi''(x_i) + \frac{h^2}{12}\phi^{(4)}(x_i) + \frac{h^4}{720}\left(\phi^{(6)}(x_i - \alpha_1 h) + \phi^{(6)}(x_i + \alpha_2 h)\right) \\
& - \frac{1}{12}\left(12\phi''(x_i) + h^2\phi^{(4)}(x_i) + \frac{h^4}{24}\left(\phi^{(6)}(x_i - \alpha_3 h) + \phi^{(6)}(x_i + \alpha_4 h)\right)\right) \\
= \;& \frac{h^4}{360}\phi^{(6)}(x_i + \alpha_5 h) - \frac{h^4}{144}\phi^{(6)}(x_i + \alpha_6 h),\ \alpha_5, \alpha_6 \in ]-1,1[.
\end{aligned}$$

Ainsi $\left|\frac{1}{h^2}\left(\phi(x_{i-1})-2\phi(x_i)+\phi(x_{i+1})\right)-\frac{1}{12}\left(\phi''(x_{i-1})+10\phi''(x_i)+\phi''(x_{i+1})\right)\right|$ peut être majoré par $\gamma h^4 \sup_{x\in[x_{i-1},x_{i+1}]}\left|\phi^{(6)}(x)\right|$.

**2.** Sachant que $u''(x)=c(x)u(x)-f(x)$, nous pouvons écrire que

$$\begin{aligned}&\frac{1}{h^2}\left(u(x_{i-1})-2u(x_i)+u(x_{i+1})\right)-\frac{1}{12}\left(u''(x_{i-1})+10u''(x_i)+u''(x_{i+1})\right)\\ =\ &\frac{1}{h^2}\left(u(x_{i-1})-2u(x_i)+u(x_{i+1})\right)\\ &-\frac{1}{12}\left(c(x_{i-1})u(x_{i-1})+10c(x_i)u(x_i)+c(x_{i+1})u(x_{i+1})-f(x_{i-1})-10f(x_i)-f(x_{i+1})\right)+erreur\end{aligned}$$

Nous en déduisons que la solution approchée $V$ devra vérifier pour $i=2,\ldots,N+1$ :

$$\begin{aligned}&\frac{1}{h^2}\left(v_{i-1}-2v_i+v_{i+1}\right)\\ =\ &-\frac{1}{12}\left(c(x_{i-1})v_{i-1}+10c(x_i)v_i+c(x_{i+1})v_{i+1}-f(x_{i-1})-10f(x_i)-f(x_{i+1})\right).\end{aligned}$$

soit en multipliant par $h^2$ et en réorganisant les 'équations :

$$(-1+h^2c(x_{i-1})v_{i-1}+(2+c(x_i))v_i+(-1+h^2c(x_{i+1})v_{i+1}=\frac{h^2}{12}(f(x_{i-1})+10f(x_i)+f(x_{i+1})).$$

Sachant que $v_1=\alpha$ et $v_{N+2}=\beta$, on retrouve le système $AW=b$ où $A$ et $b$ ont été définis précédemment.

Par continuité de la fonction $c$, la matrice $A$ est à nouveau à diagonale dominante stricte pour $N$ assez grand et donc inversible si bien que le système précédent admet une solution unique.

**3...6.** `numerov4.m`

```
N=5
alpha=1;
beta=-1;
h=1/(N+1);h2=h*h;
X=(0:h:1)';
C=c(X);
F=f(X);
A1=2*diag(ones(N,1),0)-diag(ones(N-1,1),1)-diag(ones(N-1,1),-1);
A2=10*diag(C(2:N+1),0)+diag(C(3:N+1),1)+diag(C(2:N),-1);
A=A1+h2/12*A2;
b=h2/12*(F(1:N)+10*F(2:N+1)+F(3:N+2));
b(1)=b(1)+(1-h2/12*C(1))*alpha;
b(N)=b(N)+(1-h2/12*C(N+2))*beta;
V=[alpha;A\b;beta];
XX=0:1/500:1;
U=u(X);
err=norm(U-V,inf)
plot(XX,u(XX),X,V,'x')
```

**7.** Comme le schéma de Numerov est construit à partir d'une approximation de la somme de dérivées secondes faisant intervenir une erreur avec la dérivée 6ième, dès qu'on prend un polynôme de degré 5 comme solution du problème exact, cette dernière coïncidera avec la solution du problème approché. Exemple : $c(x) = x$, $u(x) = 1 - 2x^4$, $f(x) = x + 24x^2 - 2x^5$.

```
>> numerov4
err =
  1.1102e-016
```

L'erreur est nulle aux erreurs machine près.

**8.** numerov5.m

```
alpha=1;beta=-1;
tN=[2,3,5,10,15,20,35,50,100]
for i=1:length(tN)
    N=tN(i)
    h=1/(N+1);h2=h*h;
    X=(0:h:1)';
    C=c(X);
    F=f(X);
    A1=2*diag(ones(N,1),0)-diag(ones(N-1,1),1)-diag(ones(N-1,1),-1);
    A2=10*diag(C(2:N+1),0)+diag(C(3:N+1),1)+diag(C(2:N),-1);
    A=A1+h2/12*A2;
    b=h2/12*(F(1:N)+10*F(2:N+1)+F(3:N+2));
    b(1)=b(1)+(1-h2/12*C(1))*alpha;
    b(N)=b(N)+(1-h2/12*C(N+2))*beta;
    V=[alpha;A\b;beta];
    U=u(X);
    taberr(i)=norm(U-V,inf);
end;
taberr
plot(log(tN+1),log(taberr))
a=polyfit(log(tN+1),log(taberr),1)
title(['pente␣de␣la␣droite␣de␣regression␣:␣'...
    ,num2str(a(1))],'FontSize',16)
xlabel('log(N+1)','FontSize',16)
ylabel('log(Erreur)','FontSize',16)
```

**9.** **10.** numerov7.m

```
N=6
alpha=1;
beta=2^((1-sqrt(5))/2);
h=1/(N+1);h2=h*h;
T=1;nt=10;
tau=T/nt;
X=(0:h:1)';
C=c1(X)
A1=2*diag(ones(N,1),0)-diag(ones(N-1,1),1)-diag(ones(N-1,1),-1);
A2=10*diag(C(2:N+1),0)+diag(C(3:N+1),1)+diag(C(2:N),-1);
B=h2/tau*1/12*(10*diag(ones(N,1),0)+diag(ones(N-1,1),1)...
```

```
    +diag(ones(N-1,1),-1));
A=A1+h2/12*A2+B;
b2=zeros(N,1);
b2(1)=(1-h2/12*C(1))*alpha
b2(N)=(1-h2/12*C(N+2))*beta
W(:,1)=alpha+(beta-alpha)*X;
for i=1:nt-1
    W(:,i+1)=[alpha;A\(B*W(2:N+1,i)+b2);beta];
end
for i=1:nt
    plot(X,W(:,i)); hold on
end
xx=0:T/40:T;
plot(xx,uinf(xx),'xr')
hold off
```

```
function y=c1(x)
y=1./(1+x).^2;
```

```
function y=uinf(x);
r=(1-sqrt(5))/2;
y=(x+1).^r;
```

## 10.4 Équation de convection-diffusion

**1.** Partant de l'équation $-u''(x_i) + rc(x_i)u'(x_i) = f(x_i)$ pour $i = 2, \ldots, N+1$, on obtient par approximation que la solution approchée $\widetilde{V}$ vérifie :

$$\frac{-v_{i-1} + 2v_i - v_{i+1}}{h^2} + rc(t_i)\frac{v_{i+1} - v_{i-1}}{2h} = f(t_i),\ i = 2, \ldots, N$$

soit en multipliant par $h^2$ et sachant que $v_1 = v_{N+2} = 0$ et $\widetilde{V} = \begin{pmatrix} 0 \\ V \\ 0 \end{pmatrix}$

$$\left(A + \frac{rh}{2}B\right) V = F$$

où $A$, $B$ et $F$ ont été définis précédemment.

**2...7.** `CD6.m`

```
clear
N=4
r=1
h=1/(N+1);
```

```
t=0:h:1;
F=h^2*f(r,t(2:N+1))';
C=c(t(2:N+1))';
A=2*diag(ones(N,1))-diag(ones(N-1,1),1)-diag(ones(N-1,1),-1);

B=diag(C)*(-diag(ones(N-1,1),-1)+diag(ones(N-1,1),1));
M=A+r*h/2*B;

V=M\F;
Vh=[0;V;0]
Uex=u(r,t)';
tt=0:1/200:1;
plot(t,Vh,tt,u(r,tt))
err=norm(Vh-Uex,inf)
title(['␣r␣=␣',num2str(r),'␣,␣N␣=␣',int2str(N)],'FontSize',16)
```

```
function y=c(x)
y=x;

function y=f(r,x)
y=r^2*exp(r*x).*(x-1)/(1-exp(r))+r*x;

function y=u(r,x);
y=x-(1-exp(r*x))/(1-exp(r));
```

**8.** CD7.m

```
tN=[5 10 20 35 50 75 100];
%tN=[40:3:60];
r=1;
for i=1:length(tN)
    N=tN(i);
    h=1/(N+1);
    t=0:h:1;
    F=h^2*f(r,t(2:N+1))';
    C=c(t)';
    A=2*diag(ones(N,1))-diag(ones(N-1,1),1)-diag(ones(N-1,1),-1);
    B=diag(C(2:N+1))*(-diag(ones(N-1,1),-1)+diag(ones(N-1,1),1));
    M=A+r*h/2*B;
    V=M\F;
    V=[0;V;0];
    Uex=u(r,t)';
    taberr(i)=norm(V-Uex,inf)
end;
a=polyfit(log(tN+1),log(taberr),1);
plot(log(tN+1),log(taberr));
title(['pente␣de␣la␣droite␣de␣regression␣:␣'...
    ,num2str(a(1))],'FontSize',16)
xlabel('log(N+1)','FontSize',16)
ylabel('log(Erreur)','FontSize',16)
```

**9.** CD8.m. On reprend CD6.m sauf

```
...
g=r*h*C(1:N+2);
a=1/2+1/2*coth(g/2)-1./g
B=diag(C(2:N+1))*(-diag(a(3:N+1),-1)+diag(2*a(2:N+1)-1)...
    +diag(1-a(2:N),1));
M=A+r*h*B;
...
```

**10.** CD9.m

```
clear
tN=[40:3:60];
r=100;
for i=1:length(tN)
    N=tN(i);
    h=1/(N+1);
    t=0:h:1;
    F=h^2*f(r,t(2:N+1))';
    C=c(t)';
    A=2*diag(ones(N,1))-diag(ones(N-1,1),1)-diag(ones(N-1,1),-1);
    g=r*h*C(1:N+2);
    a=1/2+1/2*coth(g/2)-1./g
    B=diag(C(2:N+1))*(-diag(a(3:N+1),-1)+diag(2*a(2:N+1)-1)...
        +diag(1-a(2:N),1));
    M=A+r*h*B;
    V=M\F;
    V=[0;V;0];
    Uex=u(r,t)';
    taberr(i)=norm(V-Uex,inf)
end;
a=polyfit(log(tN+1),log(taberr),1);
plot(log(tN+1),log(taberr));
title(['pente␣de␣la␣droite␣de␣regression␣:␣'...
    ,num2str(a(1))],'FontSize',16)
xlabel('log(N+1)','FontSize',16)
ylabel('log(Erreur)','FontSize',16)
```

# 11 Problèmes

## ÉNONCÉS DES PROBLÈMES

### 11.1 Différences finies en dimension 2

Soient $\Omega$ un ouvert connexe de frontière $\Gamma$, $f$ et $c$ deux fonctions définies sur $\Omega$, $g$ une fonction définie sur $\Gamma$. On considère le problème suivant : trouver une fonction $u$ définie sur $\bar{\Omega}$ vérifiant

$$(\mathcal{P}) \qquad \begin{cases} -\Delta u + cu = f \quad \text{sur } \Omega \\ u\,|_{\Gamma} = g \end{cases}$$

Il s'agit d'un problème physique régi par une loi de diffusion. Deux exemples peuvent nous amener à ce problème.

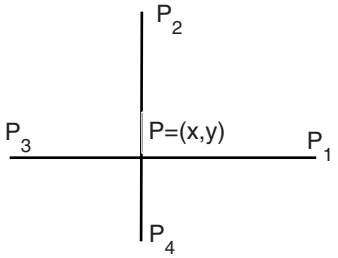
**Exemple 1 :** On considère une plaque mince et souple s'appuyant sur un contour dont la projection sur un plan horizontal est $\Gamma$. Cette plaque est soumise à l'action d'une force verticale $f(x)dx$ par élément de surface et on cherche la déflexion verticale $u$. La fonction $c$ est liée aux caractéristiques du matériau.

**Exemple 2 :** On cherche un potentiel scalaire $\psi$ dans un semi-conducteur. La densité de charge supposée connue est $\rho = q(p - n + c)$, où $q$ est la charge de l'électron, $n$ la concentration d'électrons de conduction, $p$ la concentration de trous et $c$ le dopage du semi-conducteur. On a alors l'équation de Poisson

$$-\Delta\psi = +\frac{1}{\varepsilon_S}\, q(p - n + c) \text{ où } \varepsilon_S \text{ est la permittivité du semiconducteur.}$$

*a) Approximation de $\Delta\phi$*

Soit $P = (x, y)$ un point de $\mathbb{R}^2$. Pour $h_i > 0$, $i = 1, 2, 3, 4$, on définit $P_1 = (x + h_1, y)$, $P_2 = (x, y + h_2)$, $P_3 = (x - h_3, y)$, $P_4 = (x, y - h_4)$.



Soit $\phi$ une fonction régulière sur un rectangle $R$ de $\mathbb{R}^2$ contenant tous les points $P_i$. On pose

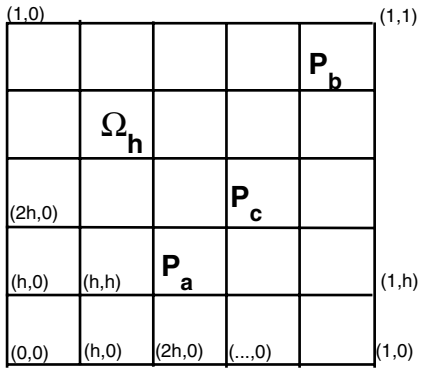
$$\Delta_h\phi(P) = \frac{2}{h_1(h_1+h_3)}\phi(P_1) + \frac{2}{h_2(h_2+h_4)}\phi(P_2) + \frac{2}{h_3(h_1+h_3)}\phi(P_3) + \frac{2}{h_4(h_2+h_4)}\phi(P_4) - \left\{\frac{2}{h_1h_3} + \frac{2}{h_2h_4}\right\}\phi(P).$$

**1.** Montrer que $|\Delta_h\phi(P) - \Delta\phi(P)| \leqslant c_1 M_3 h$ où $M_3 = \sup_{(x,y)\in\mathbb{R}^2} \left|\dfrac{\partial^3\phi}{\partial x_i^3}\right|$ (il s'agit des dérivées troisièmes dans les 2 directions, $x_1 = x$, $x_2 = y$) et $h = \max h_i$.

**2.** Si $h_1 = h_2 = h_3 = h_4 = h$, écrire la nouvelle formule $\Delta_h\phi$ et majorer $|\Delta_h\phi(P) - \Delta\phi(P)|$.

*b) Problème approché sur le carré* $\Omega = [0,1]^2$

Soit $N$ un entier strictement positif et $h = 1/(N+1)$. On définit un maillage régulier de $\Omega$ en posant $(x_i, y_j) = (ih, jh)$ pour $i, j = 0, \ldots, N+1$. Soit $\Omega_h$ l'ensemble des points du maillage situés dans $\Omega$. $\Omega_h$ est donc de cardinal $N^2$.




On cherche une approximation $U_h$ de $u$ aux points de $\Omega_h$ en supposant que $U_h$ coïncide avec $u$, donc $g$ sur le bord $\Gamma$. Le problème approché s'écrit

$$(\mathcal{P}_h) \qquad \begin{cases} -\Delta_h U_h + h^2 c U_h = h^2 f_h \text{ sur } \Omega_h \\ U_h(P) = g(P) \text{ si } P \in \Gamma \end{cases}$$

**1.** Écrire l'équation correspondante pour les points $P_a(2h, h)$, $P_b(Nh, Nh)$, $P_c(3h, 2h)$....

**2.** Montrer que le problème $(\mathcal{P}_h)$ s'écrit $A_h U_h = F_h + G_h$ où $A_h \in \mathbb{R}^{N^2\times N^2}$, $F_h$, $G_h \in \mathbb{R}^{N^2}$.

**3.** Plus précisément, montrer que

$$A_h = \begin{pmatrix} D_1 & -Id & & & \\ -Id & D_2 & -Id & & O \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & -Id \\ O & & & -Id & D_n \end{pmatrix} \text{ où } D_i = \begin{pmatrix} d_{i1} & -1 & & O \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & -1 \\ O & & -1 & d_{in} \end{pmatrix} \in \mathbb{R}^{N\times N}$$

$d_{ij} = 4 + h^2 c(ih, jh)$
$F_{hij} = h^2 f(ih, jh)$
$G_h$ est un vecteur creux correspondant aux points voisins de $\Gamma$ où $u$ est connue.

**4.** Montrer que $A_h$ est à diagonale strictement dominante dès que l'on suppose $c > 0$.

**5.** À l'aide des disques de Gershgorin, montrer que $A_h$ est inversible. Il n'est pas interdit de revenir à (4.2) du chapitre 4.

On admet que si $u \in \mathcal{C}^4(\bar{\Omega})$ alors

$$\|u - u_h\|_\infty = \max_{P\in\Omega_h} |u(P) - U_h(P)| \leqslant c_1 h^2 M_4$$

où $c_1$ est une constante et $M_4 = \sup_{(x,y)\in\bar{\Omega}} \left|\dfrac{\partial^4 \phi}{\partial x_i^4}\right|$. Cette inégalité peut être démontrée en utilisant une technique similaire à celle de l'étude d'erreur en dimension 1. Nous montrerons numériquement que $\|u - U_h\|_\infty = O(h^2)$.

Reste à résoudre le système en évitant de construire la très grande matrice $A_h$ qui comporte beaucoup de zéros (matrice creuse). Par la méthode itérative de Gauss-Seidel (cf [18] par exemple), pour résoudre $AX = b$ avec $A \in \mathbb{R}^{d\times d}$, on initialise $X^0$. Si $X^p = (x_1^p, \ldots, x_d^p)^T$ désigne le vecteur obtenu à l'étape $p$, le passage de l'étape $p$ à $p+1$ est donné par :

$$x_k^{p+1} = \frac{1}{a_{kk}} \left( b_k - \sum_{j=1}^{k-1} a_{kj}\, x_j^{p+1} - \sum_{j=k+1}^{d} a_{kj} x_j^p \right) \text{ pour } k = 1 \text{ à } d.$$

Pour résoudre $(\mathcal{P}_h)$, nous utiliserons cette méthode en profitant de la structure très creuse et très originale de $A_h$. Le vecteur inconnu $U_h$ est un vecteur à $N^2$ composantes que nous noterons $u_h(i, j)$, $i, j = 1, \ldots, N$. Nous ajoutons

$$u_h(0, j) = g(0, jh),\ u_h(N+1, j) = g(1, jh)$$
$$u_h(i, 0) = g(ih, 0),\ u_h(i, N+1) = g(ih, 1),$$

si bien que $Uh$ devient un vecteur à $(N+2)^2$ composantes, mais nous ne l'écrivons pas sous forme de vecteur.

Montrer que le passage de l'étape $p$ à l'étape $p+1$ se fait par

pour $i = 1, N$
  pour $j = 1, N$
    $v = \dfrac{1}{d(i,j)}(F_h(i,j) + u_h(i-1,j) + u_h(i,j-1) + u_h(i+1,j) + u_h(i,j+1))$
    $err(i,j) = |v - u_h(i,j)|$
    $u_h(i,j) = v$
  fin $j$
fin $i$

Lorsque $i = 1$ par exemple, $u_h(i-1,j)$ sera la contribution de $G_h$ au second membre. Il reste à itérer jusqu'à ce que $\|err\|_\infty = \max|err(i,j)| < \epsilon$ où $\epsilon$ est une précision requise entre 2 itérations ou que le nombre maximum d'itérations soit dépassé (auquel cas la méthode diverge).

### *c) Programmation*

Pour $N$ donné

**1.** À l'aide de `.m` fonctions, on construit $d(i,j)$ et $F_h(i,j)$ pour $i, j = 1, \ldots, N$,

**2.** On initialise $u_h(i,j)$ pour $i, j = 0, \ldots, N$ avec,

$$\begin{aligned} u_h(0,j) &= g(0,jh),\ u_h(N+1,j) = g(1,jh),\ u_h(i,0) = \ldots \\ u_h(i,j) &= 0 \text{ pour } i, j = 1, \ldots, N \end{aligned}$$

**3.** On fixe $maxiter$ et $precis$.

**4.** On lance la méthode de Gauss-Seidel.

**5.** Si la méthode converge, on a la solution de $\mathcal{P}_h$ dans $u_h$.

*difficulté supplémentaire :* Matlab n'accepte pas les indices à partir de 0 ; il faut donc tout décaler de 1 dans les 2 directions.

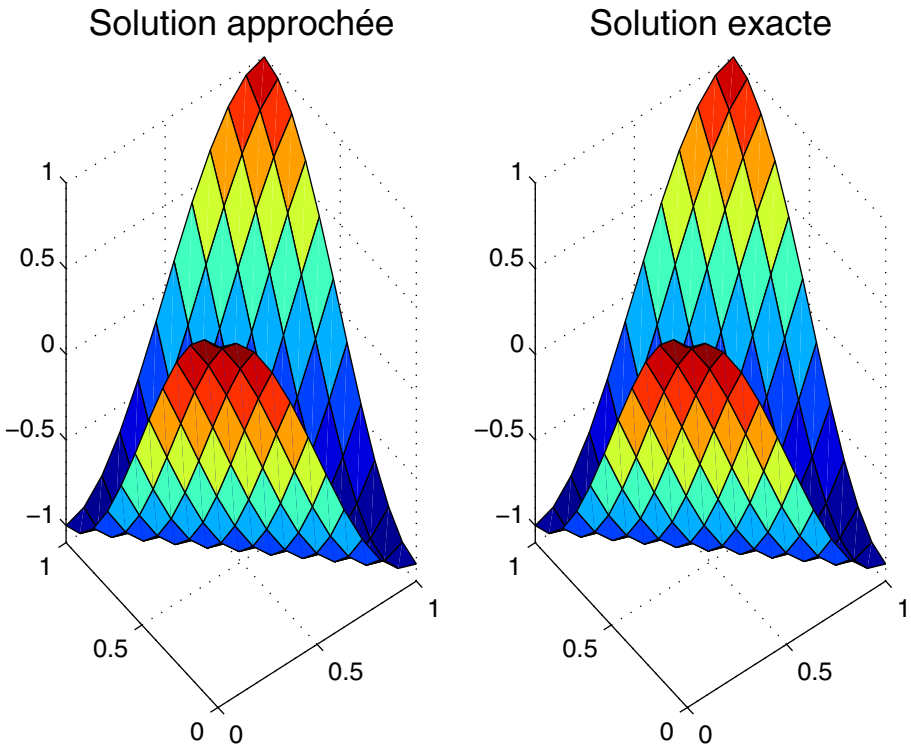
Programmer la méthode pour un $N$ donné. Valider le programme avec un exemple où $\Delta_h u = \Delta u$, c'est-à-dire que la solution approchée coïncide avec la solution exacte. On construira une fonction `diffini.m` qui étant donné $N$ renvoie le maillage $x, y$ (utiliser `meshgrid`), la solution exacte $ue$ et la solution approchée $uh$. Pour la boucle qui porte sur 2 paramètres, on pourra fixer $maxiter$ à $100N$ et $\epsilon$ à $10^{-10}$ par exemple. La boucle peut se faire en utilisant l'instruction `while`. On peut afficher $erriteration$ qui mesure l'erreur entre 2 itérations, soit $\|err\|_\infty$ et $maxiter - iter$ qui doit rester positif si on veut la convergence.

Exemple : $c(x,y) = 2\pi^2$, $f(x,y) = 4\pi^2 \sin \pi(x+y)$, la solution exacte est $u(x,y) = \sin \pi(x+y)$ qui définit aussi la fonction $g$, au bord.

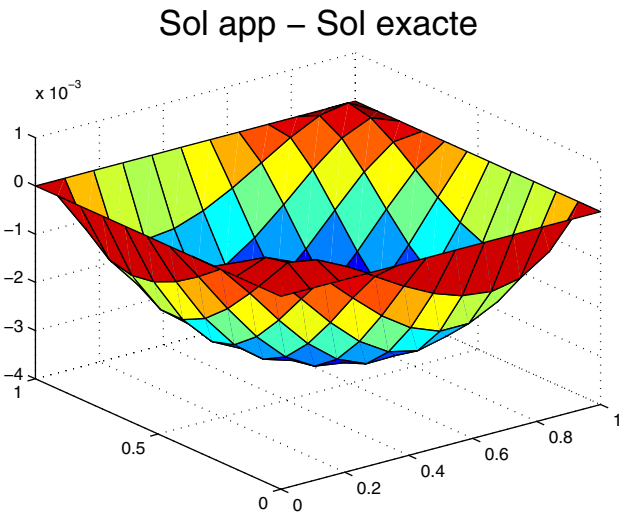
```
function [x,y,ue,uh]=diffini(n);
% x,y maillage, ue, solution exacte, uh, solution approchee
% num\'erotation des points
%     *(1,n+2)---------------*(n+2,n+2)
%     |                       |
%     |                       |
%     |            *(i,j)     |
%     |
%     *(1,2)--*(2,2)---       *(n+2,2)
%     |       |               |
%     *(1,1)--*(2,1)---       *(n+2,1)
%
%  inconnues uh(i,j) approximation de u[(i-1)*h,(j-1)*h], i,j=2,n+1
```

Rajouter un dessin de la solution approchée.

```
>> plaque
n =     10
erriteration =   9.8289e-011
maxiter-iter =    747
||sol ex-sol app|| =   0.0034
erreur*(n+1)^2 =    0.4087
```

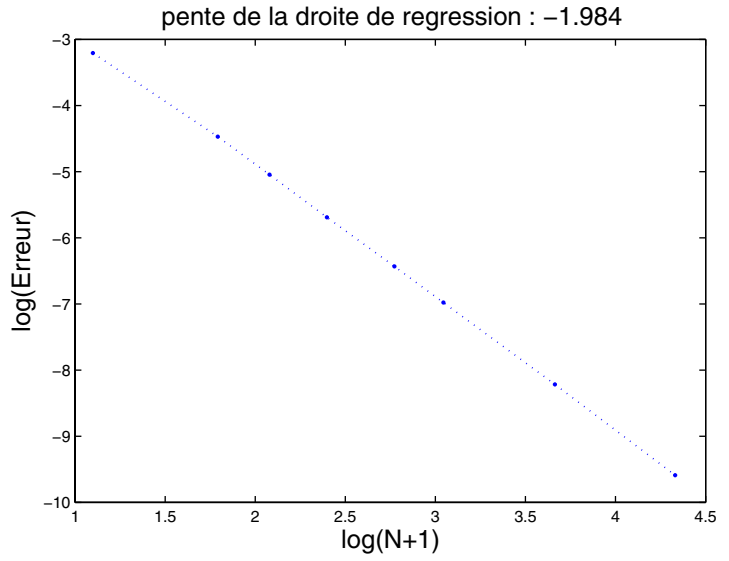


Pour le dessin ci-dessous, on notera le changement d'échelle verticale.




En faisant varier $N$, étudier $\|u - u_h\|_\infty * (N+1)^2$ ?

```
tN =
   2   5   7  10  15  20  38  75
taberreur =
  0.0405  0.0114  0.0064  0.0034  0.0016  0.0009  0.0003  0.0001
erreur*(N+1)^2 =
  0.3641  0.4122  0.4118  0.4087  0.4114  0.4105  0.4110  0.3955
```



## 11.2 Équation de la chaleur

On considère une barre de longueur $L$ portée par un axe dans laquelle la chaleur se transmet par conduction. On suppose que la barre a des échanges de chaleur avec l'extérieur qui proviennent de sources disposées tout le long ; la quantité de chaleur fournie au point $x$ à l'instant $t$ ayant une densité $f(x,t)$. La température $\theta(x,t)$ vérifie alors

$$\sigma \frac{\partial \theta}{\partial t}(x,t) - \gamma \frac{\partial^2 \theta}{\partial x^2}(x,t) = f(x,t)$$

où $\sigma$ est la chaleur spécifique linéaire et $\gamma$ la conductibilité calorifique. On suppose connue la température initiale $\theta(x,0)$ et les températures des extrémités $\theta(0,t) = \alpha(t)$ et $\theta(L,t) = \beta(t)$.

Par des changements de variables convenables, on peut supposer $\sigma = \gamma = L = 1$ et $\alpha(t) = \beta(t) = 0$. Le problème à résoudre est alors

$$(P) \begin{cases} \dfrac{\partial u}{\partial t}(x,t) - \dfrac{\partial^2 u}{\partial x^2}(x,t) = f(x,t), & 0 \leqslant x \leqslant 1, 0 \leqslant t \\ u(x,0) = u_0(x), & 0 \leqslant x \leqslant 1, \qquad \text{conditions initiales} \\ u(0,t) = u(1,t) = 0, & 0 \leqslant t, \qquad \text{conditions aux bords.} \end{cases}$$

On suppose que $u_0(0) = u_0(1) = 0$ pour des raisons de compatibilité et que $f$ et $u_0$ sont suffisamment régulières. Dans ce cas $(P)$ admet une solution unique notée $u$.

### *a) Solution approchée*

Nous allons chercher une solution approchée $v$ définie sur un maillage en espace et en temps. Pour un entier $N$ strictement positif donné, on pose $h = \frac{1}{N+1} = \Delta x$, c'est le pas d'espace et on choisit $k = \Delta t$ le pas de temps. On souhaite que $v$ soit définie en $x_i = ih, t_n = n\Delta t$ pour $i = 0$ à $N+1$ et $n \geqslant 0$. Nous noterons $V(i,n) = v(x_i,t_n)$. Nous prendrons naturellement $V(i,0) = u_0(x_i)$, $V(0,n) = V(N+1,n) = 0$ pour les conditions initiales et aux limites. Il reste à déterminer $V(i,n)$ pour $i = 1,\ldots,N$ et $n \geqslant 1$.

**1.** Soit $\phi$ une fonction définie sur $\mathbb{R}^2$ et régulière.

Montrer que

$$\frac{\partial \phi}{\partial t}(x,t) = \frac{\phi(x,t+\Delta t) - \phi(x,t)}{\Delta t} + c_1 \Delta t \frac{\partial^2 \phi}{\partial t^2}(x, t+\alpha\Delta t) \tag{11.1}$$

$$\frac{\partial \phi}{\partial t}(x,t) = \frac{\phi(x,t) - \phi(x,t-\Delta t)}{\Delta t} - c_1 \Delta t \frac{\partial^2 \phi}{\partial t^2}(x, t-\beta\Delta t) \tag{11.2}$$

$$\frac{\partial^2 \phi}{\partial x^2}(x,t) = \frac{\phi(x-h,t) - 2\phi(x,t) + \phi(x+h,t)}{h^2} + c_2 h^2 \frac{\partial^4 \phi}{\partial x^4}(x+\gamma h, t) \tag{11.3}$$

**2.** À l'aide de ces approximations des dérivées partielles, nous souhaitons, connaissant $(V(j,n))_{j=1,\ldots,N}$, déterminer $(V(i,n+1))_{i=1,\ldots,N}$ dans les 3 cas suivant :

(a) **Schéma explicite**. On utilise les formules (11.1) et (11.3) pour la fonction $u$ au point $(x_i, t_n)$. Déterminer alors $V(i, n+1)$ en fonction des $V(j, n)$, $j = i-1, i, i+1$. Écrire sous forme vectorielle le vecteur $V(:, n+1)$ en fonction de $V(:, n)$ en utilisant la matrice

$$A = \begin{pmatrix} 2 & -1 & & O \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & -1 \\ O & & -1 & 2 \end{pmatrix} \in \mathbb{R}^{N \times N}.$$

(b) **Schéma implicite**. On utilise les formules (11.2) et (11.3) pour la fonction $u$ au point $(x_i, t_{n+1})$. Déterminer une équation liant $V(i-1, n+1)$, $V(i, n+1)$, $V(i+1, n+1)$ et $V(i, n)$. Écrire sous forme vectorielle la relation entre $V(:, n+1)$ et $V(:, n)$. Montrer que connaissant $V(:, n)$ on peut ainsi déterminer $V(:, n+1)$ de manière unique (utiliser les propriétés de $A$ vues au chapitre 10).

(c) **Schéma de Crank-Nicolson**. Dans les 2 schémas précédents, on a trouvé 2 équations du type $V(i, n+1) =$ ..... On fait la moyenne des 2 formules pour obtenir une nouvelle équation puis un système liant $V(:, n+1)$ à $V(:, n)$. Écrire cette nouvelle formule et montrer que connaissant le vecteur $V(:, n)$, elle permet de déterminer un unique vecteur $V(:, n+1)$.

**3.** **Ordre des méthodes** : on reprend les 3 schémas de la question précédente, on note $u$ la solution exacte que l'on suppose régulière et $U(i, n) = u(x_i, t_n)$. Majorer les erreurs commises dans les 3 schémas en remplaçant la solution $V(i, n)$ par $U(i, n)$ en fonction de $\Delta t$, $h$ et les maxima des dérivées de $u$. Ainsi pour le schéma explicite, on évalue l'erreur

$$U(i, n+1) - U(i, n) + \frac{\Delta t}{h^2}(-U(i-1, n) + 2U(i, n) - U(i+1, n)) - \Delta t\, f(x_i, t_n).$$

Comparer les majorations pour les 3 méthodes en supposant que $h = 0(\Delta t)$, i.e. les pas de temps et d'espace sont du même ordre. Nous avons alors évalué l'erreur locale de discrétisation.

Si on résout l'équation sur l'intervalle de temps $[0, T]$, ces erreurs s'ajoutent, le nombre de pas est $T/_{\Delta t} = 0(1/_{\Delta t})$. L'ordre de la méthode est la puissance de $\Delta t$ (ou $h$) à l'arrivée. Donner l'ordre des 3 méthodes.

**4.** **Exemple :** on prend $f \equiv 0$. En développant $u_0$ en série de Fourier, on obtient une combinaison de sin $p\pi x$ et cos $p\pi x$. On va prendre $u_0(x) = \sin p\pi x$.

(a) Montrer que la solution exacte du problème $(P)$ est $u(x, t) = \sin p\pi x\, e^{-p^2\pi^2 t}$.

(b) On rappelle la formule trigonométrique

$$\sin[(i-1)\theta] - 2\sin(i\theta) + \sin[(i+1)\theta] = -4\sin^2\frac{\theta}{2}\sin(i\,\theta).$$

En déduire que les solutions approchées des 3 schémas sont de la forme $V(i, n) = \alpha^n \sin(p\pi i h)$ où $\alpha$ est une constante qui dépend du schéma, de $h$ et $\Delta t$.

(c) Déterminer d'éventuelles conditions sur $h$ et $\Delta t$ pour que la solution approchée reste bornée. Dans ce cas la méthode est dite **stable**, comme elle est d'ordre supérieur à 1, elle est convergente.

### *b) Programmation*

On part de l'équation de la chaleur simplifiée

$$(P)\ \begin{cases} \dfrac{\partial u}{\partial t}(x,t) - \dfrac{\partial^2 u}{\partial x^2}(x,t) = 0, & 0 \leqslant x \leqslant 1\ , 0 \leqslant t \leqslant T, \\ u(x,0) = u_0(x), & 0 \leqslant x \leqslant 1, \\ u(0,t) = u(1,t) = 0, & 0 \leqslant t. \end{cases}$$

pour en chercher des solutions approchées.

Soit $h = \Delta x = 1/(N+1)$ le pas d'espace et $\Delta t$ le pas de temps. On cherche des approximations de $u(ih, n\Delta t)$, $i = 0$ à $N+1$, $n = 0, 1, \ldots$ En fait $u(0, n\Delta t)$, $u(1, n\Delta t)$ et $u(i\Delta x, 0)$ étant connues, on cherche une approximation $V(i,n) \simeq u((i-1)\Delta x, (n-1)\Delta t)$, pour $i = 2$ à $N+1$ et $n = 2, \ldots$ avec la condition initiale $V(i,1) = u((i-1)h, 0) = u_0((i-1)h)$.

On choisit la méthode d'Euler implicite, soit

$$V(i,1) = u_0((i-1)h),\ i = 2 \text{ à } N+1$$
$$\left( Id + \frac{\Delta t}{(\Delta x)^2} A \right) V(2 : N+1, n+1) = V(2 : N+1, n),\ n \geqslant 1$$

où $A = \begin{pmatrix} 2 & -1 & & 0 \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & -1 \\ 0 & & -1 & 2 \end{pmatrix} \in \mathbb{R}^{N \times N}$, $Id$ est la matrice identité

et $V(2 : N+1, n) = (V(2,n), V(3,n), \ldots, V(N+1,n))^T \in \mathbb{R}^N$.

Si on prend $u_0(x) = \sin \pi x$, alors, la solution exacte de $(P)$ est $u(x,t) = \sin(\pi x) e^{-\pi^2 t}$. On prendra $T = 0.4$.

**1.** Écrire un premier programme `EQC1.m` qui, à partir de $N$, calcule et affiche $V(:,1)$, calcule et affiche la matrice $A$. Test pour $N = 6$.

```
>> EQC1
ans =
         0
    0.4339
    0.7818
    0.9749
    0.9749
    0.7818
    0.4339
         0
```

```
A =
     2    -1     0     0     0     0
    -1     2    -1     0     0     0
     0    -1     2    -1     0     0
     0     0    -1     2    -1     0
     0     0     0    -1     2    -1
     0     0     0     0    -1     2
```

**2.** Compléter le premier programme dans `EQC2.m` pour calculer et afficher $V(:,2)$ à partir de $N$ et $\Delta t$. Test $N = 6$, $\Delta t = 0.02$.

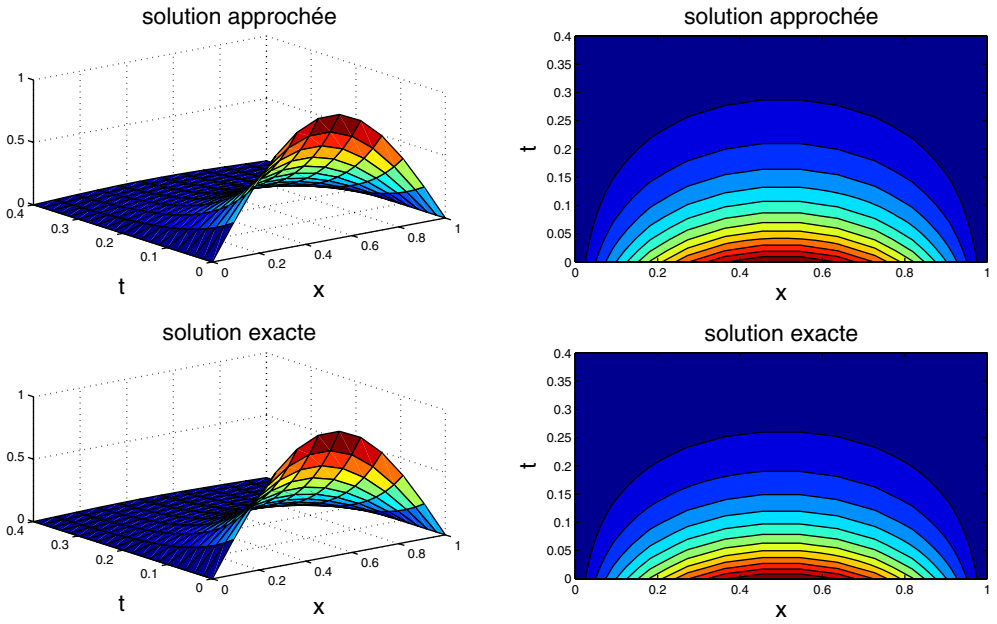
```
>> EQC3
ans =
         0
    0.3634
    0.6547
    0.8165
    0.8165
    0.6547
    0.3634
         0
```

**3.** Compléter le second programme pour calculer les vecteurs $V(:,k)$ puis dessiner les solutions des problèmes exact et approché en dimension 3. On tracera aussi les lignes de niveaux. Utiliser `meshgrid, surf, contour`. Sauvegarde `EQC4.m`. Test pour $N = 6$ et $\Delta t = 0.1$ puis $N = 10$ et $\Delta t = 0.02$.

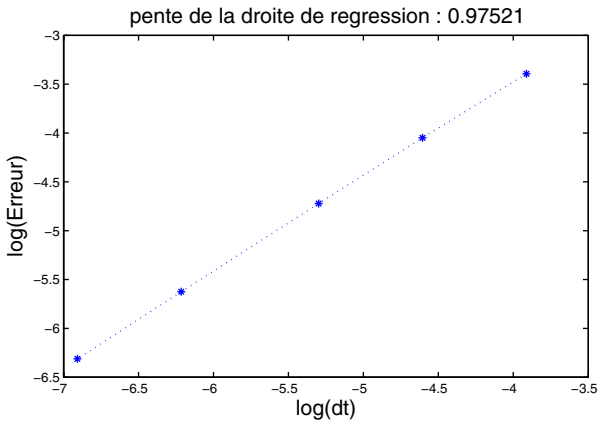
$N = 6$ et $\Delta t = 0.1$

```
>> EQC4
V =
         0         0         0         0         0
    0.4339    0.2202    0.1117    0.0567    0.0288
    0.7818    0.3968    0.2014    0.1022    0.0519
    0.9749    0.4948    0.2511    0.1274    0.0647
    0.9749    0.4948    0.2511    0.1274    0.0647
    0.7818    0.3968    0.2014    0.1022    0.0519
    0.4339    0.2202    0.1117    0.0567    0.0288
         0         0         0         0         0
```

$N = 10$ et $\Delta t = 0.02$

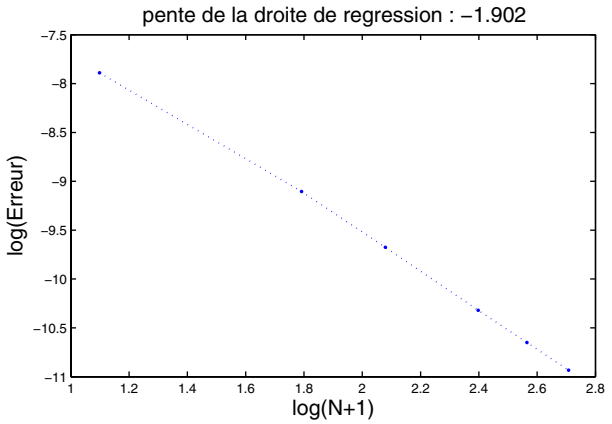



**4.** On choisit $N = 200$. Étudier l'erreur commise entre la solution exacte et la solution approchée quand $\Delta t$ varie. On pourra partir d'un tableau $tabdt = [0.02, 0.01, 0.005, 0.002, 0.001]$ et représenter $\log(erreur)$ et fonction de $\log(\Delta t)$. Sauvegarde dans `EQC5.m`.




On constate que l'erreur est de l'ordre de $\Delta t$.

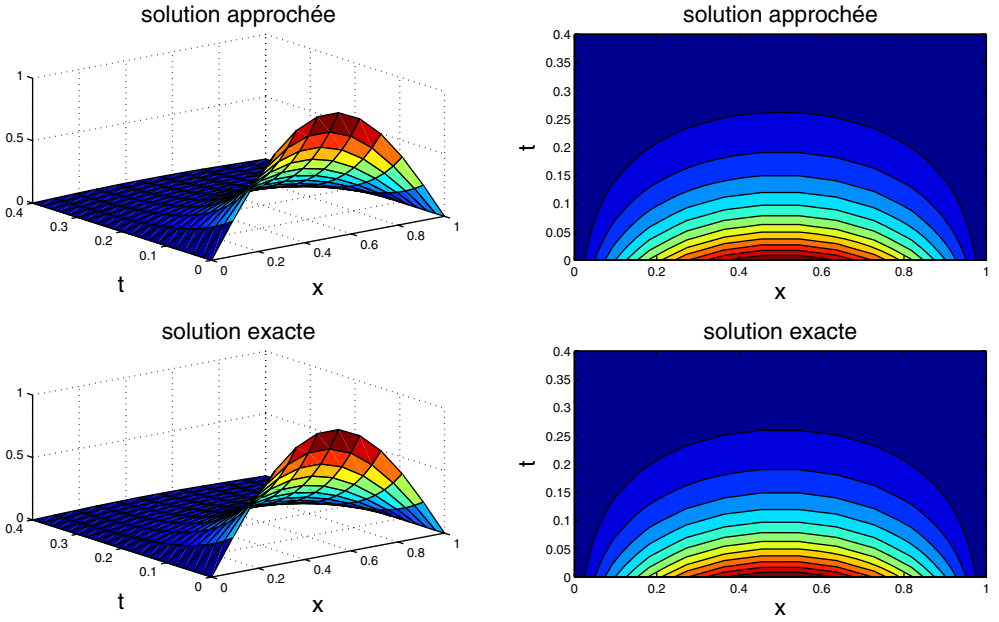
**5.** On choisit $\Delta t = 0.000001$ et $T = 0.0005$. Étudier l'erreur commise entre la solution exacte et la solution approchée quand $N$ varie. On pourra partir d'un tableau $tabN = [2, 5, 7, 10, 12, 14]$ et représenter $\log(erreur)$ et fonction de $\log(N + 1)$. Sauvegarde dans `EQC6.m`.



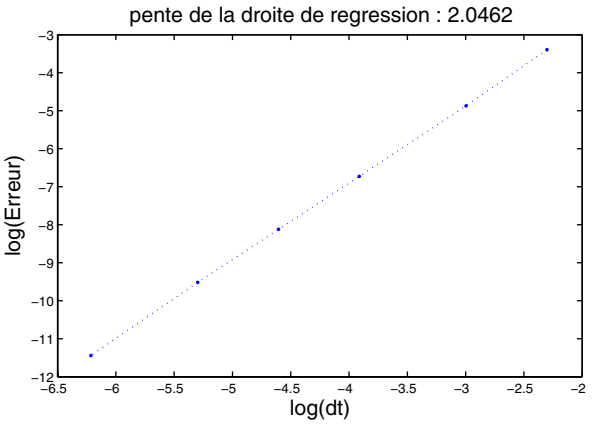

On constate que l'erreur est en $1/(N+1)^2$. Mais encore faut-il prendre $\Delta t$ très petit.

**6.** Reprendre les questions 1 à 4 pour le schéma de Crank-Nicolson

>> `CN4`




>> `CN5` avec $N = 500$ et $tabdt = [0.1, 0.05, 0.02, 0.01, 0.005, 0.002]$.




L'erreur est en $(\Delta t)^2$.

## 11.3 Éléments finis élémentaires

*a) Problème variationnel et problème approché*

On considère le problème

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

où $f$ et $c$ sont des fonctions continues avec $c > 0$. Ce problème admet une solution unique de classe $\mathcal{C}^2$ sur $[0, L]$.

**1.** Soit $v \in \mathcal{C}^1$ telle que $v(0) = v(1) = 0$, montrer que

$$\int_0^L u'(x)v'(x)dx + \int_0^L c(x)u(x)v(x)dx = \int_0^L f(x)v(x)dx. \tag{11.4}$$

Il s'agit de la formulation variationnelle du problème $(P)$.

On désigne par $V$ l'espace vectoriel des fonctions $v$ continues sur $[0, L]$ admettant des dérivées premières continues sauf en un nombre fini de points où les dérivées ont une limite à gauche et à droite. On impose de plus que $v(0) = v(L) = 0$.

**2.** Montrer que pour tout $v \in V$, (11.4) est encore vérifiée.

Si $w_1, \ldots, w_n$ sont $n$ fonctions indépendantes de $V$, on désigne par $V_n$ le sous-espace vectoriel engendré par ces fonctions. Pour le problème approché, il s'agit de déterminer $\bar{u} \in V_n$ tel que

$$\forall \bar{v} \in V_n,\ \int_0^L \bar{u}'(x)\bar{v}'(x)dx + \int_0^L c(x)\bar{u}(x)\bar{v}(x)dx = \int_0^L f(x)\bar{v}(x)dx. \tag{11.5}$$

**3.** En écrivant $\bar{u}(x) = \sum_{j=1}^{n} z_j w_j(x)$, montrer que le problème approché revient à trouver un vecteur $z = (z_1, \ldots, z_n)^T \in \mathbb{R}^n$ tel que pour $i = 1, \ldots, n$,

$$\sum_{j=1}^{n} (r_{ij} + m_{ij}) z_j = b_i,$$

où $r_{ij} = \int_0^L w_i'(x) w_j'(x) dx$, $m_{ij} = \int_0^L c(x) w_i(x) w_j(x) dx$ et $b_i = \int_0^L f(x) w_i(x) dx$.
$R = (r_{ij})$ est appelé matrice de rigidité et $M = (m_{ij})$ matrice de masse.

**4.** En écrivant $z^T(R + M)z$ sous la forme d'une intégrale, montrer que si $z^T(R + M)z = 0$ alors $z = 0$. En déduire que $R + M$ est inversible et que le problème approché admet ainsi une solution unique.

**5.** On prend $w_j(x) = \sin(j\omega x)$ où $\omega = \pi/L$. Déterminer $R$ et dans le cas où $c = 1$, déterminer $M$.

**6.** Soit $0 = x_0 < x_1 < \ldots < x_n < x_{n+1} = L$ un subdivision de $[0, L]$. Pour $i = 1, \ldots, n$, $w_i$ est la fonction continue, affine par morceaux telle que $w_i(x_j) = \delta_{ij}$ pour $j = 0, \ldots, n + 1$.

(a) Montrer que $R$ et $M$ sont tridiagonales.

(b) Dans le cas d'une subdivision régulière i.e. $x_{i+1} - x_i = h = \dfrac{L}{n+1}$, déterminer $R$. Si de plus $c = 1$ déterminer $M$.

(c) Finalement on remplace le second membre $B = (b_1, \ldots, b_n)$ où $b_i = \int_0^L f(x) w_i(x) dx$ par une approximation $F = (f_1, \ldots, f_n)$ où $f_i = \int_0^L \bar{f}(x) w_i(x) dx$ avec $\bar{f}(x) = \sum_{j=1}^{n} f(x_j) w_j(x)$.
Écrire le système donnant la solution du problème approché dans ce cas.

### *b) Programmation*

On part de l'équation de la chaleur simplifiée

$$\begin{aligned} \frac{\partial u}{\partial t}(x,t) - \frac{\partial^2 u}{\partial x^2}(x,t) &= f(x,t), \quad x \in [0,1],\ t \in [0,T] \\ u(0,t) = u(L,0) &= 0 \\ u(x,0) &= \sin(\pi x) + x^2(1-x) \end{aligned}$$

avec $f(x,t) = -2 + 6x$. On utilise une méthode d'éléments finis, $h = \dfrac{1}{N+1}$, $x_i = ih$ pour $i = 0, \ldots N + 1$. Avec les fonctions chapeau $w_i$, $i = 1$ à $N$ définie sur la subdivision régulière

(cf. ci-dessus), la solution approchée est donnée par $v(x,t) = \sum_{j=1}^{N} \eta_j(t) w_j(x)$ où

$$(E) \ : \ M\eta'(t) + R\eta(t) = MF, \text{ où } \eta = (\eta_1, \ldots, \eta_N)^T.$$

$$\text{avec } M = \frac{h}{6}\begin{pmatrix} 4 & 1 & 0 & & \ldots & 0 \\ 1 & 4 & 1 & 0 & & \\ 0 & 1 & 4 & 1 & 0 & \vdots \\ & & \ddots & \ddots & \ddots & 0 \\ \vdots & & 0 & 1 & 4 & 1 \\ 0 & \ldots & & 0 & 1 & 4 \end{pmatrix} \text{ et } R = \frac{1}{h}\begin{pmatrix} 2 & -1 & 0 & & \ldots & 0 \\ -1 & 2 & -1 & 0 & & \\ 0 & -1 & 2 & -1 & 0 & \vdots \\ & & \ddots & \ddots & \ddots & 0 \\ \vdots & & 0 & -1 & 2 & -1 \\ 0 & & \ldots & 0 & -1 & 2 \end{pmatrix}$$

Pour approcher la solution de $(E)$, on utilise une méthode de résolution de système différentiel fournie par `Matlab`, par exemple `ode23` en considérant le système $\eta'(t) = -M^{-1}R\eta(t) + F$ avec $\eta(0) = U_0 = (u(x_1,0), \ldots, u(x_N,0))^T$.

**1.** À partir de $N$, construire $M$, $R$ et $F$ (qui est constant). Sauvegarder dans `ELF1.m`, test avec $N = 4$, afficher $M$, $R$ et $F$.

```
>> ELF1
M =
    0.1333    0.0333         0         0
    0.0333    0.1333    0.0333         0
         0    0.0333    0.1333    0.0333
         0         0    0.0333    0.1333

R =
    10    -5     0     0
    -5    10    -5     0
     0    -5    10    -5
     0     0    -5    10

F =
   -0.8000
    0.4000
    1.6000
    2.8000
```

**2.** Construire une fonction `Z=phi(t,Y)` définie par $Z = -M^{-1}RY + F$. Attention, les variables $R$, $M^{-1}$ et $F$ pourront être déclarées comme variable globale à l'aide de l'instruction `global R F invM` contenue dans le programme `ELF1.m` et rappelée dans les fonctions les utilisant ; on initialise ces valeurs dans le programme `ELF1.m`. Sauvegarder dans `phi.m`. Test `phi(1,[0 :4]')` après avoir modifié `ELF1.m`.

```
>> phi(1,[1:4]')
ans =
    2.7885
```

```
  -13.9541
   55.4278
 -198.1569
```

**3.** Construire une fonction `u0.m` qui permet l'initialisation. Test `u0((0 :0.2 :1)'`.

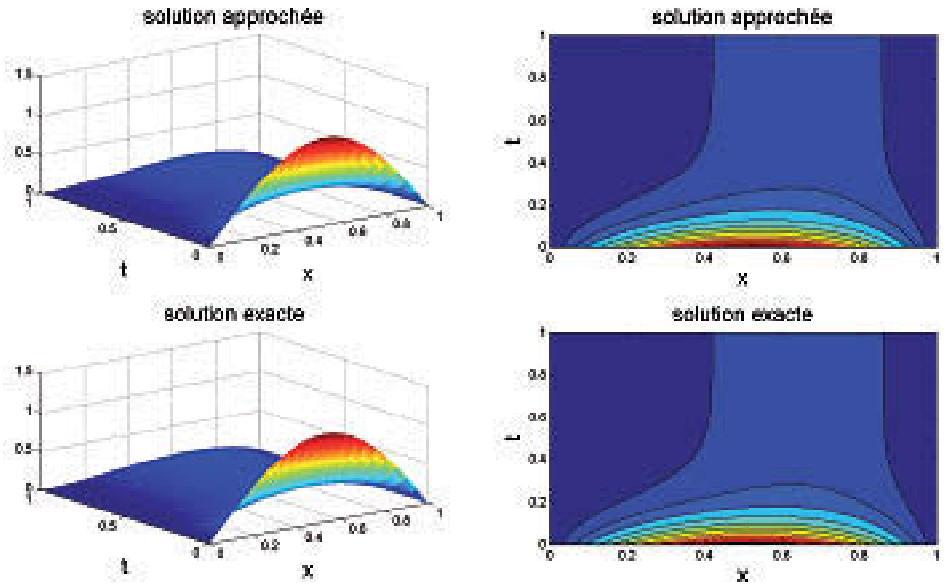
```
>> u0((0:0.2:1)')
ans =
         0
    0.6198
    1.0471
    1.0951
    0.7158
    0.0000
```

**4.** Compléter le premier programme pour calculer la solution approchée $V$ en utilisant l'instruction `ode23` puis dessiner les solutions des problèmes exact et approché en dimension 3. Solution exacte : $u(x,t) = e^{-\pi^2 t} \sin \pi x + x^2(1-x)$. On tracera aussi les lignes de niveaux. Utiliser `meshgrid, surf, contour`. Sauvegarder `ELF2.m`. Test pour $N = 4$, afficher $V(3,:)$, puis graphique avec $N = 20$.

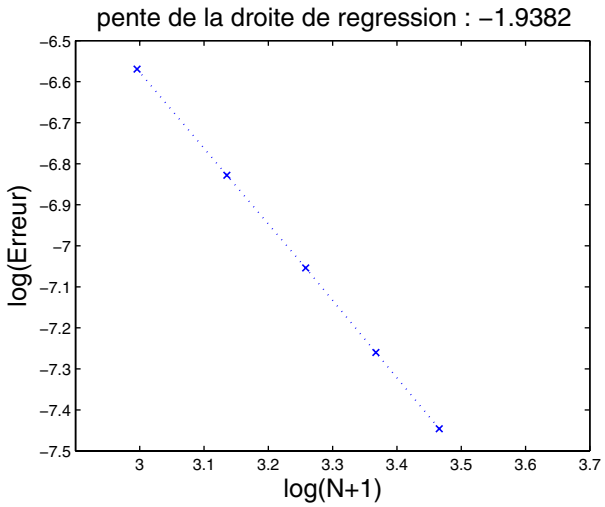
$N = 4$

```
>> V(3,:)
ans =
         0    0.5256    0.8882    0.9334    0.6073         0
```

$N = 20$



**5.** Comparer la solution approchée à la solution exacte et étudier l'erreur quand $N$ varie $T = 1$. On partira de $tN = [20 : 3 : 32]$ et on évaluera l'erreur au temps final, $t = 1$. Sauvegarder sous `ELF3.m`




La méthode est d'ordre 2 en espace.

## CORRIGÉS DES PROBLÈMES

### 11.1 Différences finies en dimension 2

*a) Approximation de $\Delta\phi$*

**1.** L'approximation de $\Delta\phi$ est à nouveau obtenue à partir d'une formule de Taylor. En effet, compte tenu des hypothèses de régularité de $\phi$, pour $i = 1, 2$, il existe $\theta_i \in ]0, 1[$ tels que

$$\begin{aligned}\phi(x+h_1,y) &= \phi(x,y) + h_1\frac{\partial\phi}{\partial x}(x,y) + \frac{h_1^2}{2}\frac{\partial^2\phi}{\partial x^2}(x,y) + \frac{h_1^3}{6}\frac{\partial^3\phi}{\partial x^3}(x+\theta_1 h_1,y)\\ \phi(x-h_3,y) &= \phi(x,y) - h_3\frac{\partial\phi}{\partial x}(x,y) + \frac{h_3^2}{2}\frac{\partial^2\phi}{\partial x^2}(x,y) - \frac{h_3^3}{6}\frac{\partial^3\phi}{\partial x^3}(x-\theta_3 h_3,y)\end{aligned}$$

Sachant que $\dfrac{2}{h_1h_3} = \dfrac{2}{h_1(h_1+h_3)} + \dfrac{2}{h_3(h_1+h_3)}$, nous obtenons alors

$$\begin{aligned}&\frac{2}{h_1(h_1+h_3)}\phi(x+h_1,y) + \frac{2}{h_3(h_1+h_3)}\phi(x-h_3,y) - \frac{2}{h_1h_3}\phi(x,y)\\ &= \frac{\partial^2\phi}{\partial x^2}(x,y) + \frac{1}{3(h_1+h_3)}\left(h_1^2\frac{\partial^3\phi}{\partial x^3}(x+\theta_1h_1,y) - h_3^2\frac{\partial^3\phi}{\partial x^3}(x-\theta_3h_3,y)\right)\end{aligned}$$

si bien que si $h_x = \max(h_1, h_3)$ et $M_{x3} = \sup_{t \in ]x-h_3, x+h_1[} \left| \frac{\partial^3 \phi}{\partial x^3}(x+t, y) \right|$, on obtient

$$\left| \frac{2}{h_1(h_1+h_3)}\phi(x+h_1, y) + \frac{2}{h_3(h_1+h_3)}\phi(x-h_3, y) - \frac{2}{h_1 h_3}\phi(x, y) - \frac{\partial^2 \phi}{\partial x^2}(x, y) \right| \leqslant \frac{1}{3(h_1+h_3)}(h_1^2 + h_3^2)M_{x3} \leqslant \frac{2}{3}h_x M_{x3}$$

car $\frac{h_i}{h_1+h_3} \leqslant 1$ pour $i = 1, 3$.
On obtient une majoration identique dans la direction $y$ si bien que :

$$|\Delta\phi(x, y) - \Delta_h\phi(x, y)| \leqslant \frac{4}{3}hM_3.$$

où $h = \max(h_1, h_2, h_3, h_4)$ et $M_3 = \max_{(u,v) \in \mathbb{R}^2} \left( \left| \frac{\partial^3 \phi}{\partial x^3}(u, v) \right|, \left| \frac{\partial^3 \phi}{\partial y^3}(u, v) \right| \right)$.

**2.** Si $h_1 = h_3 = h$, on peut reprendre le calcul précédent mais avec un développement de Taylor à l'ordre 4. Pour $i = 1, 2$, il existe $\theta_i' \in ]0, 1[$ tels que

$$\begin{aligned} \phi(x+h, y) &= \phi(x, y) + h\frac{\partial \phi}{\partial x}(x, y) + \frac{h^2}{2}\frac{\partial^2 \phi}{\partial x^2}(x, y) + \frac{h^3}{6}\frac{\partial^3 \phi}{\partial x^3}(x, y) + \frac{h^4}{24}\frac{\partial^4 \phi}{\partial x^4}(x+\theta_1' h, y) \\ \phi(x-h, y) &= \phi(x, y) - h\frac{\partial \phi}{\partial x}(x, y) + \frac{h^2}{2}\frac{\partial^2 \phi}{\partial x^2}(x, y) - \frac{h^3}{6}\frac{\partial^3 \phi}{\partial x^3}(x, y) + \frac{h^4}{24}\frac{\partial^4 \phi}{\partial x^4}(x-\theta_3' h, y) \end{aligned}$$

alors

$$\begin{aligned} &\frac{\phi(x+h, y) + \phi(x-h, y) - 2\phi(x, y)}{h^2} \\ &= \frac{\partial^2 \phi}{\partial x^2}(x, y) + \frac{h^2}{24}\left( \frac{\partial^4 \phi}{\partial x^4}(x+\theta_1' h, y) + \frac{\partial^4 \phi}{\partial x^4}(x-\theta_3' h, y) \right) \\ &= \frac{\partial^2 \phi}{\partial x^2}(x, y) + \frac{h^2}{12}\frac{\partial^4 \phi}{\partial x^4}(x+\alpha h, y), \end{aligned}$$

en remarquant que par le théorème des valeurs intermédiaires, puisque $\phi^{(4)}$ est continue, $\frac{\partial^4 \phi}{\partial x^4}(x+\theta_1' h, y) + \frac{\partial^4 \phi}{\partial x^4}(x-\theta_3' h, y) = 2\frac{\partial^4 \phi}{\partial x^4}(x+\alpha h, y)$ avec $\alpha \in [-\theta_3', \theta_1'] \subset ]-1, 1[$. C'est exactement le même résultat que celui que nous avions en dimension 1 (cf (10.5)).
Nous en déduisons que, si pour $i = 1, 2, 3, 4$, $h = h_i$, en définissant

$$\Delta_h\phi(x, y) = \frac{\phi(x+h, y) + \phi(x-h, y) + \phi(x, y+h) + \phi(x, y-h) - 4\phi(x, y)}{h^2},$$

nous avons $|\Delta\phi(x, y) - \Delta_h\phi(x, y)| \leqslant \frac{h^2}{6}M_4$.

### *b) Problème approché sur le carré*

**1.** Au point $P_a(2h, h)$, nous avons

$$\Delta_h U_h(2h,h) = \frac{U_h(3h,h) + U_h(h,h) + U_h(2h,2h) + U_h(2h,0) - 4U_h(2h,h)}{h^2}.$$

Sachant que $U_h(2h, 0) = g(2h, 0)$, nous obtenons l'équation :

$$\big(4 + h^2 c(2h,h)\big)\, U_h(2h,h) - U_h(3h,h) - U_h(h,h) - U_h(2h,2h) = h^2 f(2h,h) + g(2h,0).$$

De même, au point $P_b(Nh, Nh)$, nous avons

$$\big(4 + h^2 c(Nh,Nh)\big)\, U_h(Nh,Nh) - U_h((N-1)h,Nh) - U_h(Nh,(N-1)h) \\ = h^2 f(Nh,Nh) + g(Nh,1) + g(1,Nh),$$

et au point $P_c(3h, 2h)$,

$$\big(4 + h^2 c(3h,2h)\big)\, U_h(3h,2h) - U_h(4h,2h) - U_h(3h,3h) - U_h(2h,2h) - U_h(3h,h) = h^2 f(3h,2h).$$

Il y a donc 2 sortes de points dans $\Omega_h$ : ceux qui sont « voisins » de la frontière, type $P_a$ ou $P_b$, pour lesquels certains des 4 voisins utilisés sont sur la frontière où $U_h$ est connue et vaut $g$, et ceux qui sont « intérieurs », type $P_c$, pour lesquels les valeurs de $U_h$ aux 4 voisins sont aussi des inconnues.

**2.** À chaque point de $\Omega_h$ correspond donc une équation linéaire et $(\mathcal{P}_h)$ peut s'écrire $A_h U_h = F_h + G_h$ où $A_h \in \mathbb{R}^{N^2 \times N^2}$ et $F_h, G_h \in \mathbb{R}^{N^2}$, $F_h$ correspond au second membre de l'équation initiale et ses composantes sont du type $h^2 f(x_i, y_j)$, $G_h$ correspond à la contribution des points frontières au calcul de $\Delta_h$.

**3.** En étudiant les équations en chaque point, ligne par ligne, de bas en haut, on obtient l'écriture du système proposée.

**4.** Pour le point $P_{ij} = (ih, jh)$ de $\Omega_h$, dans l'équation correspondante, le coefficient diagonal de $A_h$ est $4 + h^2 c(ih, jh)$. Sur la même ligne, en dehors de la diagonale, il y a au plus 4 fois $-1$ et donc $A_h$ est à diagonale dominante stricte dès que $c > 0$, ce qui est une hypothèse.

**5.** Les valeurs propres $\lambda$ de $A_h = (a_{k\ell})_{k,\ell=1,\dots,N^2}$ sont dans au moins un des $N^2$ disques de Gershgorin $\left\{ z \in \mathbb{C}, |z - a_{kk}| \leqslant \sum_{\ell=1,\ \ell\neq k}^{N^2} |a_{k\ell}| \right\}_{k=1,\dots,N^2}$, si bien que 0 ne peut être une valeur propre. Donc $A_h$ est inversible et le problème $(\mathcal{P}_h)$ admet une solution unique.

### *c) Programmation*

La validation du programme se fait en prenant $u$ une fonction polynômiale du second degré, ainsi $\Delta_h u = \Delta_u$ et solutions exacte et approchée coïncident. On choisit $c > 0$ et on construit $f$ pour que $-\Delta u + cu = f$.

```
function [x,y,ue,uh]=diffini(n);
h=1/(n+1);h2=h*h;
% initialisation
[x,y]=meshgrid((0:n+1)*h);
d(1:n+2,1:n+2)=4+h2*c(x,y);
fh(1:n+2,1:n+2)=h2*f(x,y);
uh=zeros(n+2);
aux=(0:n+1)*h;
uh(:,1)=u_ex(aux,0)';
uh(:,n+2)=u_ex(aux,1)';
uh(1,:)=u_ex(0,aux);
uh(n+2,:)=u_ex(1,aux);
% resolution du systeme par une methode iterative
maxiter=100*n;precis=1e-10;
iter=0;err=1;
er=zeros(n+2);
v=uh;
while (err>precis)&(iter<maxiter)
   iter=iter+1;
   v(2:n+1,2:n+1)=(fh(2:n+1,2:n+1)+uh(1:n,2:n+1)...
       +uh(3:n+2,2:n+1)+uh(2:n+1,1:n)...
       +uh(2:n+1,3:n+2))./d(2:n+1,2:n+1);
   er=abs(uh-v);
   uh=v;
   err=max(max(er));
end;
erriteration=err
disp('maxiter-iter');
maxiter-iter
ue(1:n+2,1:n+2)=u_ex(x,y);
```

### Ce qu'il faut retenir de cet exercice

1. La commande `Matlab [x,y]=meshgrid(((0 :n+1)*h)` crée deux tableaux $x$ et $y$ de même dimension (cf. chapitre 1).

2. Noter aussi l'utilisation de `while` pour obtenir une boucle avec deux conditions d'arrêt (cf. Corrigé 2.3).

```
function z=c(x,y);
z=2*pi^2;

function z=f(x,y);
z=4*pi^2*cos(pi*(x+y));

function z=u_ex(x,y);
z= cos(pi*(x+y));
```

plaque.m

```
clear, close all
n=10
[x,y,ue,uh]=diffini(n);
er=abs(ue-uh);
disp('||sol␣ex-sol␣app||')
err=max(max(er))
disp('erreur*(n+1)^2')
err*(n+1)^2
subplot(1,2,1)
surf(x,y,uh)
title('Solution␣approchee','Fontsize',14)
axis([0 1 0 1 -1.1 1])
subplot(1,2,2)
surf(x,y,ue)
title('Solution␣exacte','Fontsize',14)
axis([0 1 0 1 -1.1 1])
figure
surf(x,y,uh-ue)
title('Sol␣app␣-␣Sol␣exacte','Fontsize',20)
figure
surf(x,y,uh,'LineStyle','none','FaceColor','red');
camlight left; lighting phong
title('Solution␣approchee','Fontsize',16)
```

### Ce qu'il faut retenir de cet exercice

Noter les instructions qui donnent un lissage des solutions :
`surf(x,y,uh,'LineStyle','none','FaceColor','red') ;`
`camlight left ; lighting phong`

etuderreur.m

```
tN=[2 5 7 10 15 20 38 75];
for k=1:length(tN)
   n=tN(k)
   [x,y,ue,uh]=diffini(n);
   er=abs(ue-uh);
   taberreur(k)=max(max(er));
   taberreur1(k)=taberreur(k)*(n+1)^2;
end
tN,taberreur
disp('erreur*(N+1)^2')
taberreur1
a=polyfit(log(tN+1),log(taberreur),1);
plot(log(tN+1),log(taberreur));
title(['pente␣de␣la␣droite␣de␣regression␣:␣'...
    ,num2str(a(1))],'FontSize',16)
xlabel('log(N+1)','FontSize',16)
ylabel('log(Erreur)','FontSize',16)
```

## 11.2 Équation de la chaleur

*a) Solution approchée*

**1.** Si $\phi$ est une fonction définie et de classe $\mathcal{C}^2$ au voisinage de $(x,t)$, pour $\Delta t$ assez petit, par la formule de Taylor, il existe $\alpha \in ]0,1[$ tel que

$$\phi(x,t+\Delta t) = \phi(x,t) + \Delta t \frac{\partial \phi}{\partial t}(x,t) + \frac{1}{2}(\Delta t)^2 \frac{\partial^2 \phi}{\partial t^2}(x, t+\alpha \Delta t)$$

formule équivalente à (11.1) avec $c_1 = -1/2$. De même pour (11.2).

Si $\phi$ est une fonction définie et de classe $\mathcal{C}^4$ au voisinage de $(x,t)$, pour $h$ assez petit, par la formule de Taylor, il existe $\gamma_1$ et $\gamma_2$ dans $]0,1[$ tels que

$$\phi(x+h,t) = \phi(x,t) + h\frac{\partial \phi}{\partial x}(x,t) + \frac{1}{2}h^2 \frac{\partial^2 \phi}{\partial x^2}(x,t) + \frac{1}{6}h^3 \frac{\partial^3 \phi}{\partial x^3}(x,t) + \frac{1}{24}h^4 \frac{\partial^4 \phi}{\partial x^4}(x+\gamma_1 h, t),$$
$$\phi(x-h,t) = \phi(x,t) - h\frac{\partial \phi}{\partial x}(x,t) + \frac{1}{2}h^2 \frac{\partial^2 \phi}{\partial x^2}(x,t) - \frac{1}{6}h^3 \frac{\partial^3 \phi}{\partial x^3}(x,t) + \frac{1}{24}h^4 \frac{\partial^4 \phi}{\partial x^4}(x-\gamma_2 h, t).$$

Sachant que par le théorème des valeurs intermédiaires, il existe $\gamma \in [-\gamma_2, \gamma_1] \subset ]-1,1[$ tel que $\frac{\partial^4 \phi}{\partial x^4}(x+\gamma_1 h,t) + \frac{\partial^4 \phi}{\partial x^4}(x-\gamma_2 h,t) = 2\frac{\partial^4 \phi}{\partial x^4}(x+\gamma h,t)$, il vient

$$\begin{aligned}
&\frac{\phi(x+h,t)+\phi(x-h,t)-2\phi(x,t)}{h^2} \\
&= \frac{\partial^2 \phi}{\partial x^2}(x,t) + \frac{h^2}{24}\left(\frac{\partial^4 \phi}{\partial x^4}(x+\gamma_1 h,t) + \frac{\partial^4 \phi}{\partial x^4}(x-\gamma_2 h,t)\right) \\
&= \frac{\partial^2 \phi}{\partial x^2}(x,t) + \frac{h^2}{12}\frac{\partial^4 \phi}{\partial x^4}(x+\gamma h,t)
\end{aligned}$$

soit (11.3) avec $c_2 = -1/12$.

**2.** (a) Pour tout $n > 0$, à partir de (11.1) et (11.3), nous obtenons pour tout $i = 1, \ldots, N$,

$$\frac{V(i,n+1)-V(i,n)}{\Delta t} - \frac{V(i-1,n)-2V(i,n)+V(i+1,n)}{h^2} = f(x_i,t_n)$$

ou encore

$$V(i,n+1) = \left(1 - 2\frac{\Delta t}{h^2}\right) V(i,n) + \frac{\Delta t}{h^2} V(i-1,n) + \frac{\Delta t}{h^2} V(i+1,n) + \Delta t f(x_i,t_n).$$

En écrivant $V(1:N,n) = (V(1,n), \ldots, V(N,n))^T$ et puisque $V(0,n) = V(N+1,n) = 0$, nous obtenons

$$V(1:N,n+1) = (Id - \frac{\Delta t}{h^2}A)V(1:N,n) + \Delta t F_n \tag{11.6}$$

où $A = \begin{pmatrix} 2 & -1 & & O \\ -1 & \ddots & \ddots & \\ & \ddots & \ddots & -1 \\ O & & -1 & 2 \end{pmatrix} \in \mathbb{R}^{N\times N}$, $Id$ est la matrice identité

et $F_n = \begin{pmatrix} f(x_1,t_n) \\ \vdots \\ f(x_N,t_n) \end{pmatrix}$.

(b) Cette fois ci, nous utilisons (11.2) et (11.3) pour $t = t_{n+1}$, nous obtenons pour tout $i = 1,\dots,N$,

$$\frac{V(i,n+1)-V(i,n)}{\Delta t} - \frac{V(i-1,n+1)-2V(i,n+1)+V(i+1,n+1)}{h^2} = f(x_i,t_{n+1})$$

soit

$$\left(1+2\frac{\Delta t}{h^2}\right)V(i,n+1) - \frac{\Delta t}{h^2}V(i-1,n+1) - \frac{\Delta t}{h^2}V(i+1,n+1) = V(i,n) + \Delta t f(x_i,t_{n+1}).$$

Finalement

$$(Id + \frac{\Delta t}{h^2}A)V(1:N,n+1) = V(1:N,n) + \Delta t F_{n+1}. \tag{11.7}$$

La matrice $(Id + \frac{\Delta t}{h^2}A)$ est à diagonale dominante stricte et est donc inversible puisque 0 n'est pas une valeur propre (cf (4.2) du chapitre 4) et donc $V(1:N,n+1)$ est déterminé de manière unique par (11.7).

(c) En faisant la moyenne de (11.6) et (11.7), il vient

$$(Id + \frac{\Delta t}{2h^2}A)V(1:N,n+1) = (Id - \frac{\Delta t}{2h^2}A)V(1:N,n) + \frac{\Delta t}{2}(F_n + F_{n+1}). \tag{11.8}$$

De même que précédemment, ce système admet une solution unique.

**3.** (a) Si $u$ est la solution du problème exact, nous définissons

$$\varepsilon_1(x,t) = u(x,t+\Delta t) - u(x,t) + \frac{\Delta t}{h^2}(-u(x-h,t) + 2u(x,t) - u(x+h,t)) - \Delta t f(x,t).$$

D'après (11.1) et (11.3), nous avons

$$\begin{aligned}\varepsilon_1(x,t) &= \Delta t\frac{\partial u}{\partial t}(x,t) + \frac{1}{2}\Delta t^2\frac{\partial^2 u}{\partial t^2}(x,t+\alpha t) \\ &\qquad + \Delta t\left(-\frac{\partial^2 u}{\partial x^2}(x,t) - \frac{h^2}{12}\frac{\partial^4 u}{\partial x^4}(x+\gamma h,t)\right) - \Delta t f(x,t) \\ &= \Delta t\left(\frac{\partial u}{\partial t}(x,t) - \frac{\partial^2 u}{\partial x^2}(x,t) - f(x,t)\right) \\ &\qquad + \Delta t\left(\frac{\Delta t}{2}\frac{\partial^2 u}{\partial t^2}(x,t+\alpha t) - \frac{h^2}{12}\frac{\partial^4 u}{\partial x^4}(x+\gamma h,t)\right)\end{aligned}$$

La première partie du membre de droite est nulle puisque $u$ est solution de $(P)$. Nous pouvons alors majorer $|\varepsilon_1(x,t)|$ par

$$|\varepsilon_1(x,t)| \leqslant \Delta t \left( \frac{\Delta t}{2} M^2_{t^2} + \frac{h^2}{12} M^4_{x^4} \right),$$

où $M^2_{t^2} = \sup \left| \dfrac{\partial^2 u}{\partial t^2} \right|$ et $M^4_{x^4} = \sup \left| \dfrac{\partial^4 u}{\partial x^4} \right|$.

Dans le cas particulier où $h = O(\Delta t)$, nous obtenons $\varepsilon_1(x,t) \leqslant C_1(\Delta t)^2$.

(b) Le calcul est similaire pour

$$\varepsilon_2(x,t) = u(x,t) - u(x,t-\Delta t) + \frac{\Delta t}{h^2}(-u(x-h,t) + 2u(x,t) - u(x+h,t)) - \Delta t f(x,t),$$

et l'erreur de consistance est du même ordre que la précédente.

(c) Nous définissons

$$\begin{aligned}
&\varepsilon_3(x,t) = u(x,t+\Delta t) - u(x,t) \\
&+\frac{\Delta t}{2h^2}(-u(x-h,t) + 2u(x,t) - u(x+h,t) - u(x-h,t+\Delta t) + 2u(x,t+\Delta t) - u(x+h,t+\Delta t)) \\
&-\frac{\Delta t}{2}(f(x,t) + f(x,t+\Delta t))
\end{aligned}$$

En utilisant un développement de Taylor en $(x,t)$ puis en $(x,t+\Delta t)$, nous avons

$$\begin{aligned}
u(x,t+\Delta t) - u(x,t) &= \Delta t \frac{\partial u}{\partial t}(x,t) + \frac{\Delta t^2}{2}\frac{\partial^2 u}{\partial t^2}(x,t) + \frac{\Delta t)^3}{6}\frac{\partial^3 u}{\partial t^3}(x,t+\alpha_1 \Delta t) \\
u(x,t+\Delta t) - u(x,t) &= \Delta t \frac{\partial u}{\partial t}(x,t+\Delta t) - \frac{1}{2}(\Delta t)^2 \frac{\partial^2 u}{\partial t^2}(x,t+\Delta t) \\
&\qquad + \frac{1}{6}(\Delta t)^3 \frac{\partial^3 u}{\partial t^3}(x,t+\alpha_2 \Delta t)
\end{aligned}$$

équations dont nous faisons la moyenne pour obtenir la première partie de $\varepsilon_3$.

Nous utilisons ensuite (11.3) en $(x,t)$ puis $(x,t+\Delta t)$ :

$$\begin{aligned}
&- u(x-h,t) + 2u(x,t) - u(x+h,t) \\
&\qquad = -h^2 \frac{\partial^2 u}{\partial x^2}(x,t) - \frac{h^4}{12}\frac{\partial^4 u}{\partial x^4}(x+\gamma h,t) \\
&- u(x-h,t+\Delta t) + 2u(x,t+\Delta t) - u(x+h,t+\Delta t) \\
&\qquad = -h^2 \frac{\partial^2 u}{\partial x^2}(x,t+\Delta t) - \frac{h^4}{12}\frac{\partial^4 u}{\partial x^4}(x+\gamma' h,t+\Delta t).
\end{aligned}$$

Il vient alors

$$\begin{aligned}\varepsilon_3(x,t) &= \frac{1}{2}\Delta t\frac{\partial u}{\partial t}(x,t) + \frac{1}{4}(\Delta t)^2\frac{\partial^2 u}{\partial t^2}(x,t) + \frac{1}{2}\Delta t\frac{\partial u}{\partial t}(x,t+\Delta t) - \frac{1}{4}(\Delta t)^2\frac{\partial^2 u}{\partial t^2}(x,t+\Delta t)\\ &+ \frac{1}{12}(\Delta t)^3\left(\frac{\partial^3 u}{\partial t^3}(x,t+\alpha_1\Delta t) + \frac{\partial^3 u}{\partial t^3}(x,t+\alpha_2\Delta t)\right)\\ &- \frac{\Delta t}{2}\left(\frac{\partial^2 u}{\partial x^2}(x,t) + \frac{h^2}{12}\frac{\partial^4 u}{\partial x^4}(x+\gamma h,t) + \frac{\partial^2 u}{\partial x^2}(x,t+\Delta t) - \frac{h^2}{12}\frac{\partial^4 u}{\partial x^4}(x+\gamma' h,t+\Delta t)\right)\\ &- \frac{\Delta t}{2}(f(x,t)+f(x,t+\Delta t)).\end{aligned}$$

Sachant que $\dfrac{\partial u}{\partial t} - \dfrac{\partial^2 u}{\partial x^2} - f = 0$ aussi bien en $(x,t)$ qu'en $(x,t+\Delta t)$, et que $\dfrac{\partial^2 u}{\partial t^2}(x,t) - \dfrac{\partial^2 u}{\partial t^2}(x,t+\Delta t) = -\Delta t\dfrac{\partial^3 u}{\partial t^3}(x,t+\alpha_3\Delta t)$, on en déduit que

$$|\varepsilon_3(x,t)| \leqslant (\Delta t)^3\frac{5}{12}M^3_{t^3} + \Delta t\frac{1}{12}M^4_{x^4}h^2.$$

Si $h = O(\Delta t)$, on obtient $|\varepsilon_3(x,t)| \leqslant C_2(\Delta t)^3$. On a gagné une puissance de $\Delta t$...
Ainsi les méthodes explicite et implicite sont d'ordre 1, celle de Crank-Nicolson est d'ordre 2.

**4.** (a) Si $u(x,t) = \sin(p\pi x)e^{-p^2\pi^2 t}$, il est facile de montrer que $\dfrac{\partial u}{\partial t} - \dfrac{\partial^2 u}{\partial x^2} = 0$. De plus $u(x,0) = \sin(p\pi x) = u_0(x)$ et $u(0,t) = u(1,t) = 0$. Donc $u$ est l'unique solution du problème $(P)$.
(b) Dans tous les schémas, $V(i,0) = \sin(p\pi x_i) = \sin(p\pi hi)$. Ensuite, par récurrence, on montre alors que

$$\begin{aligned}&\text{schéma explicite} && V_1(i,n) = q_1 V(i,0) && \text{où } q_1 = 1-\zeta\\ &\text{schéma implicite} && V_2(i,n) = q_2 V(i,0) && \text{où } q_2 = 1 - \frac{\zeta}{1+\zeta}\\ &\text{schéma Cranc-N} && V_3(i,n) = q_3 V(i,0) && \text{où } q_3 = 1 - \frac{\zeta}{1+\zeta/2}\end{aligned}$$

où $\zeta = 4\dfrac{\Delta t}{h^2}\sin^2(p\pi h/2)$.
La solution approchée reste bornée quand $n$ tend vers $+\infty$ dès que $|q_i| \leqslant 1$. Or $q_i < 1$ et $q_2, q_3 \geqslant -1$. Les méthodes implicite et de Crank-Nicolson sont inconditionnellement stables i.e. indépendamment du choix de $h$ et $\Delta t$.
Par contre $-1 \leqslant q_1 \Leftrightarrow 1/2 \geqslant \dfrac{\Delta t}{h^2}\sin^2(p\pi h/2)$ et par exemple pour $p = N+1$, on trouve la condition de stabilité $1/2 \geqslant \dfrac{\Delta t}{h^2}$.

*b) Programmation*

EQC4.m

```
clear
N=10;dt=0.02;
h=1/(N+1);
r=dt/h^2;
T=0.4;
x=0:h:1;
t=0:dt:T;
V(1:N+2,1)=u(x,0)';
V(1,1:length(t))=0;
V(N+2,1:length(t))=0;
A=2*diag(ones(N,1))-diag(ones(N-1,1),1)-diag(ones(N-1,1),-1);
M=eye(N)+r*A;
for n=1:(length(t)-1)
         V(2:N+1,n+1)=M\V(2:N+1,n);
end;
%V
[xx,tt]=meshgrid(x,t);
subplot(2,2,1)
surf(xx,tt,V')
xlabel('x','Fontsize',15)
ylabel('t','Fontsize',15)
title('solution␣approchée','Fontsize',15)
axis([0 1 0 0.4 0 1])
subplot(2,2,2)
contourf(xx,tt,V',12)
xlabel('x','Fontsize',15)
ylabel('t','Fontsize',15)
title('solution␣approchée','Fontsize',15)
uu=u(x,t);
subplot(2,2,3)
surf(xx,tt,uu')
xlabel('x','Fontsize',15)
ylabel('t','Fontsize',15)
title('solution␣exacte','Fontsize',15)
axis([0 1 0 0.4 0 1])
subplot(2,2,4)
contourf(xx,tt,uu',12)
xlabel('x','Fontsize',15)
ylabel('t','Fontsize',15)
title('solution␣exacte','Fontsize',15)
```

EQC5.m

```
N=200;
h=1/(N+1);
x=0:h:1;
T=0.4;
tabdt=[0.02,0.01,0.005,0.002,0.001];
for i=1: length(tabdt)
    dt=tabdt(i);
    r=dt/h^2;
```

```
    t=0:dt:T;
    V(1:N+2,1)=u(x,0)';
    V(1,1:length(t))=0;
    V(N+2,1:length(t))=0;
    A=2*diag(ones(N,1))-diag(ones(N-1,1),1)-diag(ones(N-1,1),-1);
    M=eye(N)+r*A;
    for n=1:(length(t)-1)
        V(2:N+1,n+1)=M\V(2:N+1,n);
    end;
    uu=u(x,t)
    taberr(i)=max(max(abs(V-uu)));
end;
plot(log(tabdt),log(taberr))
a=polyfit(log(tabdt),log(taberr),1)
title(['pente␣de␣la␣droite␣de␣regression␣:␣'...
    ,num2str(a(1))],'FontSize',16)
xlabel('log(dt)','FontSize',16)
ylabel('log(Erreur)','FontSize',16)
```

CN4.m

```
N=10;dt=0.02;
h=1/(N+1);
r=dt/(2*h^2);
T=0.4;
x=0:h:1;
t=0:dt:T;
V(1:N+2,1)=u(x,0)';
V(1,1:length(t))=0;
V(N+2,1:length(t))=0;
A=2*diag(ones(N,1))-diag(ones(N-1,1),1)-diag(ones(N-1,1),-1);
M1=eye(N)-r*A;
M2=eye(N)+r*A;
for n=1:(length(t)-1)
        V(2:N+1,n+1)=M2\ (M1*V(2:N+1,n));
end;
% reprendre ensuite EQC4.m...
```

## 11.3 Éléments finis élémentaires

### *a) Problème variationnel et problème approché*

**1.** On suppose que $u$ est une fonction de classe $\mathcal{C}^2$ sur $[0, L]$ vérifiant $-u''(x)+c(x)u(x) = f(x)$. Si on multiplie cette expression par $v(x)$ où $v$ est continue, puis qu'on l'intègre sur $[0, L]$, il vient $-\int_0^L u''(x)v(x)dx + \int_0^L c(x)u(x)v(x)dx = \int_0^L f(x)v(x)dx$. En supposant que $v$ est de classe $\mathcal{C}^1$, on peut intégrer par parties la première intégrale pour obtenir

$$\int_0^L u'(x)v'(x)dx - \left[u'(x)v(x)\right]_0^L + \int_0^L c(x)u(x)v(x)dx = \int_0^L f(x)v(x)dx.$$

Si de plus $v(0) = v(1) = 0$ alors le crochet disparaît et

$$\int_0^L u'(x)v'(x)dx + \int_0^L c(x)u(x)v(x)dx = \int_0^L f(x)v(x)dx.$$

**2.** Si $v$ est $\mathcal{C}^1$ par morceaux, on intègre par parties sur chacun des morceaux et la formule reste la même.

**3.** Si $\bar{u} = \sum_{j=1}^{n} z_j w_j$ où $\{w_1, \ldots, w_n\}$ est une base de $V_n$, par linéarité, $\bar{u}$ vérifie (11.5) si et seulement si l'équation est vérifiée pour une base de $V_n$ i.e. pour tout $i$ de 0 à $n$,

$$\begin{aligned}
&\int_0^L \bar{u}'(x)w_i'(x)dx + \int_0^L c(x)\bar{u}(x)w_i(x)dx = \int_0^L f(x)w_i(x)dx \\
\Leftrightarrow\quad &\sum_{j=1}^{n} z_j \left( \int_0^L w_j'(x)w_i'(x)dx + \int_0^L c(x)w_j(x)w_i(x)dx \right) = \int_0^L f(x)w_i(x)dx \\
\Leftrightarrow\quad &\sum_{j=1}^{n} z_j(r_{ij} + m_{ij}) = b_i,
\end{aligned}$$

où $r_{ij} = \int_0^L w_i'(x)w_j'(x)dx$, $m_{ij} = \int_0^L c(x)w_i(x)w_j(x)dx$ et $b_i = \int_0^L f(x)w_i(x)dx$. Notons que la matrice $R + M$ est symétrique.

**4.**
$$\begin{aligned}
z^t(R+M)z &= \sum_{i=1}^{n}\sum_{j=1}^{n} z_j(r_{ij} + m_{ij})z_i \\
&= \int_0^L \left( \sum_{i=1}^{n}\sum_{j=1}^{n} w_j'(x)z_j w_i'(x)z_i \right) dx + \int_0^L c(x) \left( \sum_{i=1}^{n}\sum_{j=1}^{n} w_j(x)z_j w_i(x)z_i \right) dx \\
&= \int_0^L \left( \sum_{k=1}^{n} w_k'(x)z_k \right)^2 dx + \int_0^L c(x) \left( \sum_{k=1}^{n} w_k(x)z_k \right)^2 dx \\
&= \int_0^L \left[ \left( \sum_{k=1}^{n} w_k'(x)z_k \right)^2 + c(x) \left( \sum_{k=1}^{n} w_k(x)z_k \right)^2 \right] dx \geqslant 0
\end{aligned}$$

Si $z^t(R + M)z = 0$, sachant que $c > 0$, puisqu'on additionne 2 termes positifs, on en déduit que $\int_0^L c(x) \left( \sum_{k=1}^{n} w_k z_k(x) \right)^2 dx = 0$. Puisque $\sum_{k=1}^{n} w_k z_k$ est continue sur $[0, L]$, son carré l'est aussi ainsi que $c \left( \sum_{k=1}^{n} w_k z_k \right)^2$ si bien que $c(x) \left( \sum_{k=1}^{n} w_k(x)z_k \right)^2 = 0$ sur $[0, L]$ et comme $c > 0$, c'est

que $\sum_{k=1}^{n} w_k(x)z_k = 0$ sur $[0, L]$ ; alors les $z_k$ sont nuls puisque les $w_k$ forment une base de $V_n$. La matrice $R+M$ est définie positive, donc inversible. En effet si $(R+M)z = 0$ alors, $z^t(R+M)z = 0$ donc $z = 0$.

**5.** Si $w_i(x) = \sin(i\omega x)$ avec $\omega = \pi/L$ alors $w_i$ est de classe $\mathcal{C}^1$ sur $[0, L]$ et $w_i(0) = w_i(L) = 0$. Par ailleurs $\{w_1, \ldots, w_n\}$ forme une famille libre de $\mathcal{C}^1$. Déterminons $r_{ij}$ et $m_{ij}$.

$$r_{ij} = \int_0^L w_i'(x)w_j'(x)dx = ij\omega^2/L^2 \int_0^L \cos(i\omega x)\cos(j\omega x)dx = \begin{cases} 0 & \text{si } i \neq j \\ (i\omega)^2/2L & \text{si } i = j, \end{cases}$$

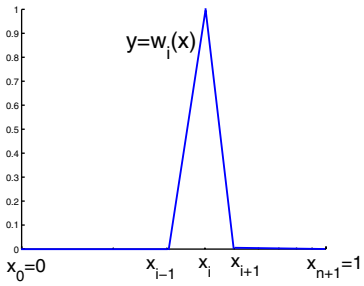
calcul classique obtenu en linéarisant le produit des fonctions trigonométriques qu'on retrouve en étudiant les séries de Fourier. La matrice $R$ est diagonale.

De même si $c(x) = 1$,

$$m_{ij} = \int_0^L w_i(x)w_j(x)dx = \int_0^L \sin(i\omega x)\sin(j\omega x)dx = \begin{cases} 0 & \text{si } i \neq j \\ L/2 & \text{si } i = j. \end{cases}$$

La matrice $M$ est aussi diagonale dès que la fonction $c$ est constante.

**6.** Sur une subdivision $x_0 = 0 < x_1 < \ldots < x_n < x_{n+1} = L$, nous définissons les fonctions chapeau $w_i$, $i = 1, \ldots, n$, continues, affines par morceaux telles que pour $j = 0, \ldots, n+1$, $w_i(x_j) = 0$. Il est clair que $w_i$ est nulle en dehors de $[x_{i-1}, x_{i+1}]$ et $\mathcal{C}^1$ par morceaux et que la dérivée vaut $\dfrac{1}{x_i - x_{i-1}}$ sur $[x_{i-1}, x_i]$ et $\dfrac{1}{x_i - x_{i+1}}$ sur $[x_i, x_{i+1}]$




Si $j \notin \{i-1, i, i+1\}$ alors $w_j$ et $w_i$ ont des supports disjoints ou ayant au plus un point commun si bien que dans ce cas $r_{ij} = m_{ij} = 0$. Ainsi les matrices $R$ et $M$ sont, au plus, tridiagonales.

Si $x_{i+1} - x_i = h$, $r_{i,i-1} = \int_{x_{i-1}}^{x_i} -1/h^2 dx = -1/h$ et de même pour les autres composantes, si bien que

$$r_{ij} = \begin{cases} 0 & \text{si } j \notin \{i-1, i, i+1\} \\ -1/h & \text{si } j = i-1, \text{ ou } j = i+1, \\ 2/h & \text{si } j = i. \end{cases}$$

De même si $c(x) = 1$, on obtient

$$m_{ij} = \begin{cases} 0 & \text{si } j \notin \{i-1, i, i+1\} \\ h/6 & \text{si } j = i-1, \text{ ou } j = i+1, \\ 2h/3 & \text{si } j = i. \end{cases}$$

Finalement, pour le second membre, on remplace $b_i = \int_0^L f(x)w_i(x)dx$ par $\bar{b}_i = \int_0^L \bar{f}(x)w_i(x)dx$ où $\bar{f}(x) = \sum_{j=0}^{n} f(x_j)w_j(x)$, si bien que $b_i = \sum_{j=0}^{n} m_{ij} f(x_j)$ puisque $c = 1$ et le système à résoudre devient :

$$(R + M)z = M\bar{b},$$

où $M = \frac{h}{6}\begin{pmatrix} 4 & 1 & 0 & & \dots & 0 \\ 1 & 4 & 1 & 0 & & \\ 0 & 1 & 4 & 1 & 0 & \vdots \\ & & \ddots & \ddots & \ddots & 0 \\ \vdots & & 0 & 1 & 4 & 1 \\ 0 & \dots & & 0 & 1 & 4 \end{pmatrix}$, $R = \frac{1}{h}\begin{pmatrix} 2 & -1 & 0 & & \dots & 0 \\ -1 & 2 & -1 & 0 & & \\ 0 & -1 & 2 & -1 & 0 & \vdots \\ & & \ddots & \ddots & \ddots & 0 \\ \vdots & & 0 & -1 & 2 & -1 \\ 0 & & \dots & 0 & -1 & 2 \end{pmatrix}$

et $\bar{b} = \begin{pmatrix} f(x_1) \\ \vdots \\ f(x_n) \end{pmatrix}$

### *b) Programmation*

`ELF2.m`

```
global R F invM
N=4
h=1/(N+1);
x=(0:h:1)';
M=h/6*(4*diag(ones(N,1))+diag(ones(N-1,1),1)+diag(ones(N-1,1),-1))
invM=inv(M);
R=1/h*(2*diag(ones(N,1))-diag(ones(N-1,1),1)-diag(ones(N-1,1),-1))
F=-2+6*x(2:N+1)
eta0=u0(x(2:N+1));
[t,V]=ode23('phi',[0,1],eta0);
V=[zeros(length(t),1),V,zeros(length(t),1)];
[xx,tt]=meshgrid(x,t);
subplot(2,2,1)
%V(3,:)
surf(xx,tt,V,'EdgeColor','none')
xlabel('x','Fontsize',18)
```

```
ylabel('t','Fontsize',18)
title('solution␣approchée','Fontsize',18)
subplot(2,2,2)
contourf(xx,tt,V,10)
xlabel('x','Fontsize',18)
ylabel('t','Fontsize',18)
title('solution␣approchée','Fontsize',18)
uu=solex(x,t');
subplot(2,2,3)
surf(xx,tt,uu','EdgeColor','none')
xlabel('x','Fontsize',18)
ylabel('t','Fontsize',18)
title('solution␣exacte','Fontsize',18)
subplot(2,2,4)
contourf(xx,tt,uu',10)
xlabel('x','Fontsize',18)
ylabel('t','Fontsize',18)
title('solution␣exacte','Fontsize',18)
```

```
function z=u0(x);
z=sin(pi*x)+x.*x.*(1-x);

function Z=phi(t,Y)
global R invM F
Z=-invM*R*Y+F;
```

### Ce qu'il faut retenir de cet exercice

L'instruction `global R invM F` est une facilité puisque les mêmes variables sont utilisées dans le programme principal et les sous programmes. Cela évite des passages en paramètre des fonctions. Le risque est de modifier les valeurs de ces variables dans différents sous programmes et de se perdre...

`ELF3.m`

```
global R F invM
tN=[20:3:32];
for i=1:length(tN)
  N=tN(i);
  h=1/(N+1);
  x=(0:h:1)';
  M=h/6*(4*diag(ones(N,1))+diag(ones(N-1,1),1)+diag(ones(N-1,1),-1));
  invM=inv(M);
  R=1/h*(2*diag(ones(N,1))-diag(ones(N-1,1),1)-diag(ones(N-1,1),-1));
  F=-2+6*x(2:N+1);
  eta0=u0(x(2:N+1));
  [t,V]=ode23('phi',[0,1],eta0);
  uu=solex(x(2:N+1),t');
  lt=length(t)
  taberr(i)=norm(V(lt,:)-uu(:,lt)',inf);
end
```

```
plot(log(tN),log(taberr))
a=polyfit(log(tN+1),log(taberr),1)
title(['pente␣de␣la␣droite␣de␣regression␣:␣'...
    ,num2str(a(1))],'FontSize',16)
xlabel('log(N+1)','FontSize',16)
ylabel('log(Erreur)','FontSize',16)
```

# Bibliographie


[1] M. Atteia, M. Pradel, *Éléments d'analyse numérique*, CEPAD, 1990.

[2] J. Bastien, *Introduction à l'analyse numérique : Applications sous Matlab*, Dunod, 2003.

[3] A. Biran, M. Breiner, *Matlab 5 for engineers*, Addison Wesley, 1999

[4] K. Chen, P. Giblin, A. Irving, *Mathematical explorations with Matlab*, Cambrige University Press, 1999.

[5] J. Cooper, *Matlab companion for multivariable calculus (a)*, Academic Press, 2001.

[6] M. Crouzeix, A. Mignot, *Analyse numérique des équations différentielles*, Masson, 1984.

[7] C. de Boor, *A practical guide to splines*, Springer-Verlag, 1978.

[8] P. Deuflhard, A. Hohmann, *Numerical Analysis in Modern Scientific Computing, An Introduction*, Springer-Verlag, 2003, 2e édition.

[9] J.-G. Dion, R. Gaudet, *Méthodes d'analyse numérique, De la théorie à l'application*, Modulo, 1996.

[10] A. Fortin, *Analyse numérique pour ingénieurs*, Ed. de l'École Polytechnique de Montréal, 1995.

[11] W. Gander, J. Hřebiček, *Solving problems in scientific computing using Maple and MATLAB*, Springer-Verlag, 2004, 4e ed.

[12] S. Godounov, V. Riabenki, *Schémas aux différences*, MIR, 1977.

[13] C. Guilpin, *Manuel de calcul numérique appliqué*, EDP Sciences, 1999.

[14] F. Gustafsson, N. Bergman, *MATLAB for engineers explained*, Springer-Verlag, 2003.

[15] F. Jerdrzejewski, *Introduction aux méthodes numériques*, Springer-Verlag, 2001.

[16] L. Jolivet, R. Labbas, *Analyse et analyse numérique : rappel de cours et exercices corrigés*, Hermès Science, 2005.

[17] A.Kharab, R. B. Guenther, *Introduction to numerical methods (a) : a Matlab approach*, Chapman and Hall, 2002.

[18] P. Lascaux et R. Théodor, *Analyse numérique matricielle appliquée à l'art de l'ingénieur*, Masson, 1986, t. 1 et 2.

[19] D. Marsh, *Applied Geometry for Computer Graphics and CAD*, Springer-Verlag, 1999.

[20] M. Mokhtari, A. Mesbah, *Apprendre à maîtriser Matlab*, Springer-Verlag, 1997 .

[21] C. B. Moler, *Numerical computing with MATLAB*, SIAM, 2004.

[22] A. Quarteroni, R. Sacco, F. Saleri *Méthodes numériques pour le calcul scientifique : Programmes en Matlab*, Springer-Verlag, 2006.

[23] J. Rappaz, M. Picasso, *Introduction à l'analyse numérique*, Presses Polytechniques et Universitaires Romandes, 1998.

[24] D. Salomon, *Curves and Surfaces for Computer Graphics*, Springer-Verlag, 2006.

[25] K. Sigmon, *Matlab Aide-Mémoire*, Springer-Verlag, 1999.

[26] E. Süli, D. Mayers, *An introduction to numerical analysis*, Cambridge Univ. Press, 2003.

[27] H. B. Wilson, L. H. Turcotte, D. Halpern, *Advanced mathematics and mechanics applications using Matlab*, Chapman and Hall, 2003, $3^e$ ed.

[28] K. Yosida, *Functional Analysis*, Springer-Verlag, 1980, $6^e$ ed.

# Index

# Index des mots clés Matlab

SCIENCES SUP

Jean-Louis Merrien

# Exercices et Problèmes
# ANALYSE NUMÉRIQUE AVEC MATLAB

Cet ouvrage se propose d'accompagner l'étudiant en Licence (Mathématiques appliquées) ou en École d'ingénieur dans son assimilation des connaissances. Dans chaque chapitre, le lecteur trouvera :

- **Un rappel de cours concis**
- **Des énoncés d'exercices et de problèmes**

Ces énoncés, en grande partie extraits de sujets d'examen, comportent des **questions détaillées et progressives**. Elles proposent une programmation sous forme de poupées russes : à chaque question, le programme précédent est amélioré et complété. **Les programmes sont téléchargeables à partir du site dunod.com**.

- **Une rubrique « Du mal à démarrer ? »**

Si le lecteur est coincé dans la résolution d'un exercice et avant d'aller voir la solution, des indications lui sont proposées pour l'aider à bien démarrer.

- **Les solutions complètes et les programmes de tous les énoncés**

**Chaque énoncé est intégralement corrigé**. Lorsque c'est utile, une rubrique « Ce qu'il faut retenir de cet exercice » propose un **bilan méthodologique**.

JEAN-LOUIS MERRIEN
est maître de conférences à l'INSA de Rennes.


MATHÉMATIQUES
PHYSIQUE
CHIMIE
SCIENCES DE L'INGÉNIEUR
INFORMATIQUE
SCIENCES DE LA VIE
SCIENCES DE LA TERRE

LICENCE 1 2 3 | MASTER 4 5 | DOCTORAT 6 7 8

6647747
ISBN 978-2-10-050863-1

www.dunod.com

DUNOD