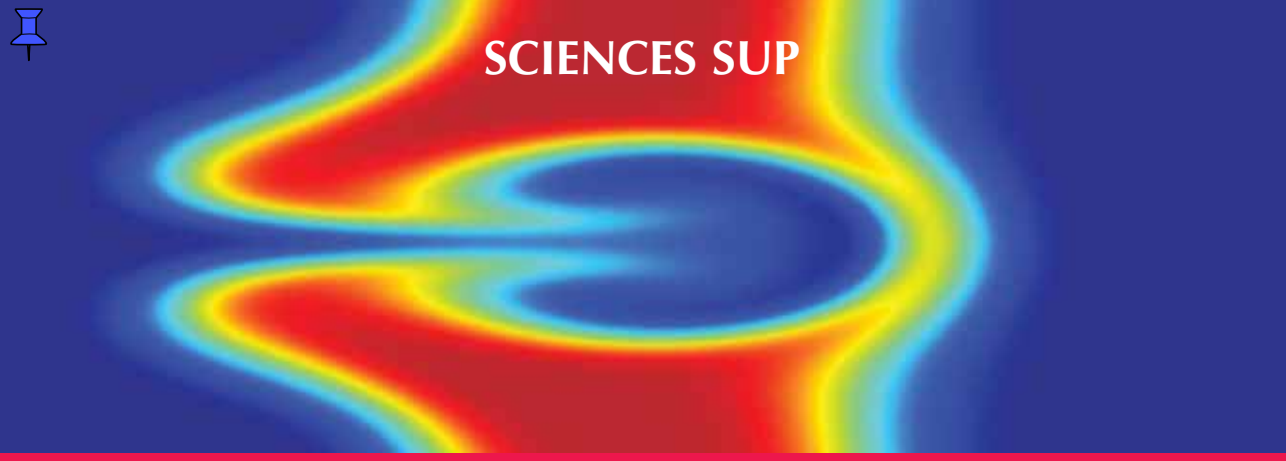


SCIENCES SUP

Exercices et problèmes corrigés avec rappels de cours

Écoles d'ingénieurs • 2e cycle/Master

INTRODUCTION AU CALCUL SCIENTIFIQUE PAR LA PRATIQUE

12 projets résolus avec MATLAB

Ionut Danaila • Pascal Joly
Sidi Mahmoud Kaber • Marie Postel

DUNOD

INTRODUCTION AU CALCUL SCIENTIFIQUE PAR LA PRATIQUE

12 projets résolus avec MATLAB

Consultez nos catalogues sur le Web

www.dunod.com

INTRODUCTION AU CALCUL SCIENTIFIQUE PAR LA PRATIQUE

12 projets résolus avec MATLAB

Ionut Danaila
Maître de conférences à l'université Paris 6

Pascal Joly
Ingénieur de recherche au CNRS

Sid Mahmoud Kaber
Maître de conférences à l'université Paris 6

Marie Postel
Maître de conférences à l'université Paris 6

DUNOD

Conseiller éditorial : Olivier Pironneau

ISBN 2 10 0048709 4

Table des matières

Avant-propos

Aujourd'hui l'enseignement des mathématiques appliquées ne peut se concevoir sans l'expérimentation numérique. En effet, seule la mise en œuvre d'une méthode sur un exemple concret permet d'en mieux comprendre les qualités (précision et rapidité de calcul) et les défauts (coût et limites d'utilisation). Bien sûr, ces éléments distinctifs peuvent être appréhendés d'un point de vue théorique ; mais la complexité de programmation et la variété des champs d'application, critères finalement déterminants pour le succès d'une méthode, ne peuvent être mesurées que par l'expérimentation.

Cet ouvrage regroupe quelques exemples d'application de méthodes désormais classiques de l'analyse numérique, mises en œuvre pour la résolution de problèmes concrets issus de la physique, la mécanique, la chimie, le traitement de l'image, etc. Ces problèmes ont été choisis pour satisfaire la curiosité légitime d'un public assez vaste. Élaborés dans le cadre de cours de mathématiques appliquées (analyse numérique, calcul scientifique), ils ont été proposés sous forme de projets à différents groupes d'étudiants de l'Université ou de Grandes Écoles d'ingénieurs.

Nous avons privilégié une approche pédagogique progressive. Chaque projet décrit précisément le problème à traiter et présente, de manière aussi complète que possible, les bases théoriques nécessaires à sa résolution. Il est structuré en étapes de difficulté croissante, pouvant être proposées de manière modulable dans le cadre d'un enseignement avec des séances de programmation étalées dans le temps. En début de projet, une "fiche" indique sa difficulté globale (sur une échelle de 1 à 3), les notions mathématiques développées, ainsi que les domaines d'application.

Toujours dans un souci pédagogique, une solution de programmation en MATLAB ® est fournie pour chaque projet. Les scripts complets peuvent être téléchargés à partir de la page web du livre, à l'adresse

`http://www.dunod.com/ICSPP/`

Dans la pratique, les méthodes de calcul scientifique étudiées sont souvent programmées dans des langages plus performants (Fortran, C, C++) adaptés à la résolution sur de gros calculateurs de cas concrets industriels. Le choix du logiciel MATLAB a de multiples motivations : la simplicité de programmation en langage interprété ; la richesse des bibliothèques pré-programmées. Enfin, la facilité de visualisation graphique des résultats. Tout ceci permet au lecteur, même néophyte en programmation, de réaliser facilement les expériences numériques proposées. Si bien que son apprentissage est un passage obligé dans la formation d'un mathématicien appliqué. Par ailleurs, ce logiciel est très largement employé aussi bien dans le monde de l'éducation que dans les laboratoires de recherche et développement. Les logiciels libres Scilab ou Octave conviennent également pour ce genre de simulation, ce qui est à prendre en compte à l'Université où l'enseignement aux niveaux Licence, Master, voire Doctorat, est souvent fait avec des logiciels libres. La traduction de nos solutions dans ces langages ne pose pas de difficulté particulière, d'autant plus que pour privilégier la lisibilité des programmes nous n'avons pas toujours cherché à utiliser les techniques de programmation conduisant à la concision maximale d'écriture.

Ionut Danaila, Pascal Joly, Sidi Mahmoud Kaber, Marie Postel.
Paris, 21 décembre 2004.

Projet **1**

Approximation numérique de quelques équations aux dérivées partielles modèles

Fiche du projet

Difficulté :	1
Notions développées :	Équations différentielles linéaires : méthodes d'intégration numérique, schémas aux différences finies : schémas d'Euler, de Runge-Kutta.
Domaines d'application :	Phénomènes de transport, diffusion, propagation d'ondes.

Le but de ce projet est de mettre en évidence les propriétés mathématiques et physiques des équations aux dérivées partielles (EDP), présentées sous la forme la plus simple possible. En fait, il ne s'agit pas d'un véritable projet, basé sur un problème bien défini, mais plutôt d'une somme d'exercices théoriques qui visent à familiariser le lecteur avec quelques techniques de base de discrétisation et d'intégration des EDP. Nous commençons par présenter ces techniques dans le cas simple des équations différentielles ordinaires (EDO), pour les étendre ensuite aux EDP modèles (équation de convection, des ondes, de la chaleur). Une attention particulière sera

accordée à l'analyse des schémas numériques (précision, stabilité, dissipation) et à la comparaison des résultats numériques avec les solutions exactes.

1.1 DISCRÉTISATION D'UNE ÉQUATION DIFFÉRENTIELLE ORDINAIRE

Considérons l'équation différentielle ordinaire (EDO) : trouver une fonction dérivable $u : [0, T] \mapsto \mathbb{R}^m$, solution de

$$u'(t) = f\big(t, u(t)\big), \tag{1.1}$$

où T est un réel strictement positif et f est une fonction continue de $[0, T] \times \mathbb{R}^m$ à valeurs dans $\mathbb{R}^m$.

Définition 1.1 *On appelle problème de Cauchy, le couplage de l'EDO (1.1) et d'une condition initiale*

$$u(0) = u_0, \tag{1.2}$$

où u_0 est un vecteur de $\mathbb{R}^m$.

Nous renvoyons à un cours sur les équations différentielles, par exemple, [Crouzeix et Mignot, 1989], [Demailly, 1996], [Delabrière et Postel, 2004], pour tout ce qui concerne l'existence et l'unicité d'une solution du problème de Cauchy (1.1)-(1.2). Dans cette première partie du projet, nous présentons des méthodes numériques simples pour calculer une approximation de la solution dans le cas scalaire (on dit aussi 1D), $m = 1$.

Puisque l'ordinateur ne peut renvoyer qu'un nombre fini de résultats, la résolution numérique du problème de Cauchy (1.1)-(1.2) va commencer par choisir les points de calcul $t_0, t_1, \ldots, t_N$ de l'intervalle $I = [0, T]$. On définit ainsi une *discrétisation* (ou un maillage) de l'intervalle I. La distribution équidistante (ou uniforme) des points de calcul est la plus simple et elle sera utilisée dans ce chapitre : l'intervalle $I = [0, T]$ est divisé en N intervalles I_n de même longueur $h = T/N$ (h est appelé pas de discrétisation). On pose $I_n = [t_n, t_{n+1}]$ avec $t_n = nh$ ($t_0 = 0, t_N = T$, voir la figure 1.1). L'approximation numérique consiste à construire

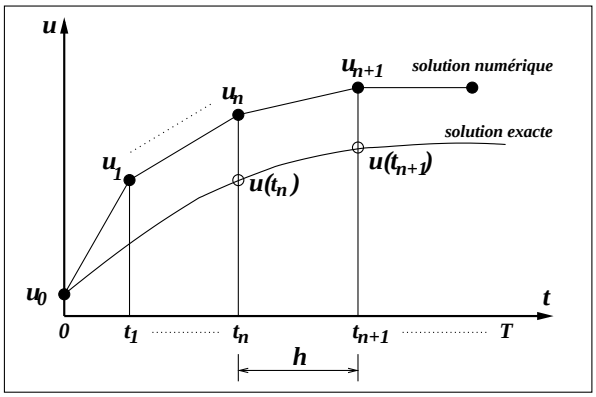


Figure 1.1 Discrétisation d'une EDO.

une suite (dépendant de N) de valeurs discrètes $u_0^{(N)}, \dots u_N^{(N)}$ approchant les valeurs $u(t_0), \dots u(t_N)$ de la solution exacte $u(t)$ aux mêmes points de calcul. On prendra toujours $u_0^{(N)} = u_0$ pour vérifier la condition initiale $u(t_0) = u_0$. Pour simplifier la présentation, on notera par la suite $u_n^{(N)}$ par u_n.

1.1.1 Construction de schémas numériques

Après la discrétisation de l'intervalle de définition I, il faut trouver une relation nous permettant de calculer les valeurs $u_n, n = 1, \dots, N$. Cette relation (*schéma numérique*) est obtenue en *discrétisant* l'opérateur différentiel intervenant dans l'EDO (rappelons que le schéma démarre de la valeur u_0, fixée par la condition initiale). On distingue deux méthodes pour construire des schémas numériques pour résoudre l'EDO (1.1).

a) Méthodes basées sur des quotients aux différences

On écrit l'équation (1.1) à l'instant t_n (la variable t désigne souvent un temps) et on remplace $u'(t_n)$ par un quotient aux différences en utilisant des développements de Taylor faisant intervenir les valeurs de l'inconnue u aux instants voisins de t_n. Prenons d'abord l'exemple de la dérivée première.

Définition 1.2 *Le pas de discrétisation h étant fixé, on définit les quotients aux différences finies*

– *décentrées en avant (ou progressives)*

$$D^+u(t) = \frac{u(t+h) - u(t)}{h}, \tag{1.3}$$

– *décentrées en arrière (ou régressives)*

$$D^-u(t) = \frac{u(t) - u(t-h)}{h}, \tag{1.4}$$

– *centrées*

$$D^0u(t) = \frac{u(t+h) - u(t-h)}{2h}. \tag{1.5}$$

Supposons que la fonction u soit deux fois continûment dérivable. Il existe alors $\theta_n^+ \in [0, h]$ tel que

$$u(t_{n+1}) = u(t_n) + hu'(t_n) + \frac{h^2}{2}u''(t_n + \theta_n^+). \tag{1.6}$$

On déduit de ce développement une approximation de $u'(t_n)$

$$u'(t_n) = \frac{u(t_{n+1}) - u(t_n)}{h} - \frac{h}{2}u''(t_n + \theta_n^+) \simeq D^+u(t_n). \tag{1.7}$$

Définition 1.3 *On dit de $D^+u(t_n)$ que c'est une approximation de $u'(t_n)$ d'ordre un car l'erreur d'approximation ε_n tend vers* 0 *comme h*

$$\varepsilon_n = \left|D^+u(t_n) - u'(t_n)\right| \leqslant h\frac{1}{2}\max_{t\in I_n}|u''(t)|. \tag{1.8}$$

De même, $D^-u(t_n) = [u(t_n) - u(t_{n-1})]/h$ est une approximation d'ordre un de $u'(t_n)$ et $D^0u(t_n) = [u(t_{n+1}) - u(t_{n-1})]/(2h)$ est une approximation d'ordre deux de $u'(t_n)$ (l'erreur d'approximation tend vers 0 comme h^2).

De manière générale, il est possible d'utiliser des combinaisons linéaires de plusieurs quotients aux différences pour trouver des approximations de $u'(t_n)$. Par exemple, on peut approcher

$$u'(t_n) \simeq \alpha D^-u(t_n) + \beta D^0u(t_n) + \gamma D^+u(t_n), \tag{1.9}$$

avec les paramètres α, β et γ choisis de sorte que l'approximation soit la plus précise possible.

Le développement de Taylor reste l'outil de base pour la construction d'approximations des dérivées d'ordre supérieur. Pour la dérivée seconde, par exemple, il suffit d'additionner le développement (1.6) poussé jusqu'à l'ordre 4, avec le développement similaire pour $u(t_{n-1})$ pour obtenir une approximation centrée (d'ordre deux) de la dérivée seconde :

$$u''(t_n) \simeq D^-D^+u(t_n) = \frac{u(t_{n+1}) - 2u(t_n) + u(t_{n-1})}{h^2}. \tag{1.10}$$

Voyons maintenant comment, à partir de ces approximations aux différences finies, construire des schémas numériques pour l'EDO (1.1). En considérant l'EDO à l'instant t_n et en remplaçant $u'(t_n)$ par $D^+u(t_n)$, on obtient le schéma

$$u_{n+1} = u_n + hf(t_n, u_n), \tag{1.11}$$

où u_{n+1} et u_n deviennent les approximations numériques de $u(t_{n+1})$, respectivement $u(t_n)$.

Le schéma (1.11) est appelé schéma d'Euler *explicite*, ou plus simplement schéma d'Euler. La méthode est dite explicite car le calcul de u_{n+1} est fait explicitement en fonction de t_n et u_n. Plus généralement, si le calcul de u_{n+1} est donné exclusivement en fonction des valeurs calculées aux instants antérieurs, $u_0, \ldots, u_n$, on dit que ce schéma est explicite.

Si on prend maintenant l'EDO (1.1) à l'instant t_{n+1} et on remplace $u'(t_{n+1})$ par $D^-u(t_{n+1})$, on obtient le schéma d'Euler *implicite* :

$$u_{n+1} = u_n + hf(t_{n+1}, u_{n+1}). \tag{1.12}$$

Cette fois le calcul de u_{n+1} implique la résolution de l'équation (1.12) qui est généralement non-linéaire.

L'approximation $u'(t_n) \simeq D^0 u(t_n)$ dans (1.1) prise à l'instant t_n conduit au schéma

$$u_{n+1} = u_{n-1} + 2hf(t_n, u_n), \tag{1.13}$$

appelé schéma Leapfrog (saute-mouton).

b) Méthodes dérivées de l'intégration numérique

La deuxième manière de construire un schéma numérique utilise les méthodes d'intégration (ou quadrature) numérique. En intégrant l'EDO (1.1) sur l'intervalle I_n, nous obtenons

$$u(t_{n+1}) - u(t_n) = \int_{t_n}^{t_{n+1}} f\big(s, u(s)\big) ds = \mathcal{I}_n. \tag{1.14}$$

On peut donc calculer $u(t_{n+1})$ connaissant $u(t_n)$, pourvu que l'on sache aussi calculer l'intégrale $\mathcal{I}_n$. On se ramène ainsi à un problème de quadrature numérique.

Plusieurs méthodes de quadrature peuvent être utilisées pour approcher l'intégrale de (1.14) :

– *la méthode des rectangles à gauche*

$$\mathcal{I}_n \simeq hf(t_n, u_n), \tag{1.15}$$

conduit au schéma d'Euler explicite (1.11) ;

– *la méthode des rectangles à droite*

$$\mathcal{I}_n \simeq hf(t_{n+1}, u_{n+1}), \tag{1.16}$$

définit le schéma d'Euler implicite (1.12) ;

– *la méthode du point milieu*

$$\mathcal{I}_n \simeq hf\Big(t_n + h/2, u(t_n + h/2)\Big), \tag{1.17}$$

qui donne, après avoir fait l'approximation (h est considéré petit),

$$u(t_n + h/2) \simeq u(t_n) + \frac{h}{2} u'(t_n) = u(t_n) + \frac{h}{2} f(t_n, u(t_n)), \tag{1.18}$$

le schéma explicite d'Euler modifié :

$$u_{n+1} - u_n = hf\left(t_n + \frac{h}{2}, u_n + \frac{h}{2} f(t_n, u_n)\right), \tag{1.19}$$

– *la méthode des trapèzes*

$$\mathcal{I}_n \simeq \frac{h}{2} \left[f(t_n, u_n) + f(t_{n+1}, u_{n+1})\right], \tag{1.20}$$

permet d'obtenir le schéma *semi-implicite* de Crank-Nicolson (voir le tableau 1.1).

1.1.2 Forme générale des schémas numériques

La forme générale d'un schéma numérique pour résoudre l'EDO (1.1) est la suivante

$$u_{n+1} = u_n + F(h; t_{n,1}, u_{n,1}, \ldots, t_{n,q}, u_{n,q}). \tag{1.21}$$

Si F dépend de q valeurs $u_{n,j}$, distinctes de u_{n+1}, on parle d'un schéma à q pas. Par exemple, le schéma Leapfrog est un schéma à deux pas. Si F ne dépend pas des instants $t_j > t_n$, le schéma est dit explicite, sinon il est dit implicite.

Les principaux schémas numériques utilisés en pratique sont groupés dans le tableau 1.1.

> **Remarque 1.1 :** Pour démarrer un schéma à un pas il suffit d'une valeur u_0 qui sera toujours prise égale à $u(0)$. La situation est différente pour les schémas à $q > 1$ pas : ces schémas ne peuvent être utilisés que pour calculer les valeurs u_n pour $n \geqslant q$ et cela suppose connues les q premières valeurs $u_0, \ldots u_{q-1}$. Comme seule la valeur u_0 est connue, pour calculer les valeurs manquantes, on utilise, par exemple, un schéma à un pas pour calculer u_1, puis un schéma à deux pas pour calculer u_2 ... et un schéma à $q-1$ pas pour calculer u_{q-1} .

Tableau 1.1 Schémas numériques pour l'EDO $u'(t) = f(t,u)$.

Euler explicite (ordre 1)	$u_{n+1} = u_n + hf(t_n, u_n)$
Euler implicite (ordre 1)	$u_{n+1} = u_n + hf(t_{n+1}, u_{n+1})$
Leapfrog (ordre 2)	$u_{n+1} = u_{n-1} + 2hf(t_n, u_n)$
Euler modifié (ordre 2)	$u_{n+1} = u_n + hf(t_n + \frac{h}{2}, u_n + \frac{h}{2} f(t_n, u_n))$
Crank-Nicolson (ordre 2)	$u_{n+1} = u_n + \frac{h}{2}[f(t_n, u_n) + f(t_{n+1}, u_{n+1})]$
Adams-Bashfort (ordre 2)	$u_{n+1} = u_n + h[\frac{3}{2} f(t_n, u_n) - \frac{1}{2} f(t_{n-1}, u_{n-1})]$
Adams-Bashfort (ordre 3)	$u_{n+1} = u_n + h[\frac{23}{12} f(t_n, u_n) - \frac{16}{12} f(t_{n-1}, u_{n-1}) + \frac{5}{12} f(t_{n-2}, u_{n-2})]$
Adams-Moulton (ordre 3)	$u_{n+1} = u_n + h[\frac{5}{12} f(t_{n+1}, u_{n+1}) + \frac{8}{12} f(t_n, u_n) - \frac{1}{12} f(t_{n-1}, u_{n-1})]$
Runge-Kutta (Heun) (ordre 2)	$\begin{cases} k_1 = hf(t_n, u_n) \\ k_2 = hf(t_n + h, u_n + k_1) \\ u_{n+1} = u_n + \frac{1}{2}(k_1 + k_2) \end{cases}$
Runge-Kutta (ordre 4)	$\begin{cases} k_1 = hf(t_n, u_n) \\ k_2 = hf(t_n + h/2, u_n + k_1/2) \\ k_3 = hf(t_n + h/2, u_n + k_2/2) \\ k_4 = hf(t_n + h, u_n + k_3) \\ u_{n+1} = u_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4) \end{cases}$

1.1.3 Application à l'étude de l'équation d'absorption

L'équation modèle pour décrire un phénomène d'absorption (ou de production) est la suivante : on cherche une fonction $u : \mathbb{R}^+ \to \mathbb{R}$ solution du problème de Cauchy

$$\begin{cases} u'(t) + \alpha u(t) = f(t), \quad \forall t > 0, \\ u(0) = u_0, \end{cases} \tag{1.22}$$

où $\alpha \in \mathbb{R}$ est donné et le terme source f prend en compte la création de la quantité u au cours du temps.

Exemple 1 : L'intensité des radiations émises par un corps radioactif est estimée en mesurant la concentration $u(t)$ d'un isotope instable. Cette concentration est divisée par deux au cours d'une durée T, appelée demi-vie, suivant la loi :

$$u'(t) = \alpha u(t), \quad \text{avec} \quad \alpha = -\frac{\ln 2}{T}.$$

Exercice 1.1

Considérons le problème de Cauchy (1.22).

1. On pose $u(t) = e^{-\alpha t} v(t)$. Écrire l'équation différentielle vérifiée par v. Résoudre cette équation et en déduire que
$$u(t) = e^{-\alpha t} \left(u_0 + \int_0^t e^{\alpha s} f(s) ds \right). \tag{1.23}$$
2. Calculer la solution dans le cas où α est une fonction de t.
3. Pour α et f constants, déterminer u et calculer $\lim_{t \to +\infty} u(t)$.
4. On prend cette fois $f = 0$, mais un coefficient plus général, $\alpha \in \mathbb{C}$, de partie réelle $\alpha_r > 0$. Montrer que $\lim_{t \to +\infty} u(t) = 0$.
5. Écrire une fonction qui met en œuvre le schéma d'Euler explicite (1.11). L'en-tête de la fonction sera :

```
function u=EulerExp(fun,u0,t0,t1,n)
% Arguments d'entrée :
%    fun  le nom de la fonction second membre de l'EDO
%    t0   le temps initial
%    u0   la condition initiale en t0
%    t1   le temps final
%    n    le nombre de pas de temps entre t0 et t1
% Argument de sortie :
%    u  le vecteur de dimension n+1 contenant la solution
%       numérique correspondant aux temps t0+i*h, avec h = (t1-t0)/n
```

(Pour la version MATLAB on utilisera la commande `feval` pour évaluer la fonction `fun` à l'intérieur de la fonction `EulerExp`.)

Dans un programme principal, appeler la fonction *EulerExp* pour résoudre l'EDO suivante : $u'(t) + 4u(t) = 0$ vérifiant la condition initiale $u(0) = 1$. On prendra $t_0 = 0, t_1 = 3$ et $n = 24$ ($h = 1/8$). Représenter sur la même figure la solution exacte et la solution numérique. Refaire le calcul pour $n = 6$ ($h = 1/2$). Commenter les résultats.

6. Utiliser à la place de la méthode d'Euler explicite la méthode Runge-Kutta d'ordre 4 (on écrira une fonction *RKutta4* suivant le modèle de la fonction *EulerExp*). Commenter les résultats pour $h = 1/2$.

La solution de cet exercice se trouve à la page 19.

Stabilité d'un schéma numérique

On considère ici le cas $f = 0$ et $\alpha \in \mathbb{R}^+$. La solution de l'équation (1.22) est alors $u(t) = e^{-\alpha t}u_0$ et $\lim_{t\to+\infty} u(t) = 0$. Supposons que l'on cherche à calculer cette solution par le schéma d'Euler explicite (1.11). On obtient une suite de valeurs $u_n = (1 - \alpha h)^n u_0$. Notons que :

– si $h > 2/\alpha$, on a $1 - \alpha h \leqslant -1$ et la suite u_n diverge,

– et si $0 < h < 2/\alpha$ on a $|1 - \alpha h| < 1$ et la suite u_n tend vers 0 comme la solution exacte.

Cette analyse explique le comportement *bizarre* de la solution numérique de la question 5 de l'exercice 1.1 pour $h = 1/2$ ($\alpha = 4$). Il s'agit d'une question fondamentale pour le comportement des schémas numériques : le paramètre $\alpha > 0$ étant donné, comment choisir le pas de discrétisation h pour que la suite des solutions approchées ait le même comportement à l'infini que la solution exacte. C'est le problème de la stabilité des schémas numériques.

Considérons maintenant le cas plus général d'un schéma à un pas, donné sous la forme :

$$u_{n+1} = u_n + hF_h(t_n, u_n). \tag{1.24}$$

En tenant compte que la solution exacte vérifie la relation

$$u(t_{n+1}) = e^{-\alpha h}u(t_n), \tag{1.25}$$

on cherchera à écrire les schémas du tableau 1.1 sous la forme

$$u(t_{n+1}) = G(-\alpha h)u(t_n). \tag{1.26}$$

Les propriétés de décroissance à l'infini de la solution numérique dépendront de la fonction G qu'on appellera *fonction d'amplification du schéma*. À titre d'exercice, le

lecteur est invité à retrouver les fonctions d'amplification suivantes (cf. tableau 1.1) :

– Euler explicite $G(z) = 1 + z$
– Euler implicite $G(z) = 1/(1 - z)$
– Euler modifié $G(z) = (2 + z)/(2 - z)$
– RKutta2 $G(z) = 1 + z + z^2/2$
– RKutta4 $G(z) = 1 + z + z^2/2 + z^3/6 + z^4/24$

Définition 1.4 *L'ensemble des points $z \in \mathbb{C}$ pour lesquels $|G(z)| < 1$ est appelé domaine de stabilité du schéma.*

Suivant la relation (1.26), on voit que si $-\alpha h$ appartient au domaine de stabilité du schéma, alors

$$\lim_{n\to+\infty} |u(t_n)| \leqslant \lim_{n\to+\infty} |u(t_0)|\ |G(-\alpha h)|^n = 0.$$

Par exemple, le domaine de stabilité du schéma d'Euler explicite est le disque ouvert du plan centré au point $(-1, 0)$ et de rayon 1. Le schéma sera donc stable si on prend un pas de discrétisation h tel que $|1 - \alpha h| < 1$.

Exercice 1.2

Tracer sur la même figure les frontières des domaines de stabilité des schémas Euler explicite, RKutta2 et RKutta4, c'est-à-dire l'ensemble des points z du plan pour lesquels $|G(z)| = 1$. On pourra se restreindre au rectangle $[-4, 1] \times [-4, 4]$. (Pour la version matlab utiliser les fonctions `meshgrid` et `contour`.)

La solution de cet exercice se trouve à la page 19.

1.2 ÉQUATIONS AUX DÉRIVÉES PARTIELLES MODÈLES

On définit de manière générale une équation aux dérivées partielles (EDP) comme la relation entre une fonction de plusieurs variables et ses dérivées partielles. Les EDP présentées dans cette partie servent à modéliser des phénomènes physiques élémentaires : la convection, la propagation des ondes et la diffusion. Pour chacun de ces phénomènes, nous présentons une ou plusieurs équations modèles, le calcul analytique des solutions ainsi que des schémas numériques pour calculer des approximations de ces solutions.

Nous aborderons dans ce projet les EDP modèles suivantes :

– l'équation de convection : $\partial_t u(x, t) + c\partial_x u(x, t) = f(x, t)$,
– l'équation des ondes : $\partial^2_{tt} u(x, t) - c^2 \partial^2_{xx} u(x, t) = 0$,
– l'équation de la chaleur : $\partial_t u(x, t) - \kappa \partial^2_{xx} u(x, t) = f(x, t)$.

1.2.1 L'équation de convection

L'équation modèle pour le phénomène de convection (ou de transport, ou encore d'advection) est la suivante : on cherche une fonction $u(x,t) : \mathbb{R} \times \mathbb{R}^+ \to \mathbb{R}$ solution de l'équation aux dérivées partielles (EDP)

$$\partial_t u(x,t) + c\partial_x u(x,t) = f(x,t), \quad \forall x \in \mathbb{R}, \quad \forall t > 0, \tag{1.27}$$

vérifiant la condition initiale

$$u(x,0) = u_0(x), \quad \forall x \in \mathbb{R}. \tag{1.28}$$

Cette équation modélise le transport d'une quantité $u(x,t)$ (dépendant de la position x et de l'instant t), connaissant la distribution initiale $u(x,0)$, ainsi que la vitesse de transport c (supposée constante) de cette quantité.

> **Exemple 2 :** Soit $u(x,t)$ la concentration au temps t et à la position x d'un polluant dans l'air. Notant par c la vitesse du vent, le transport du polluant est gouverné par l'équation $\partial_t u + \partial_x(cu) = 0$, qui est bien du type (1.27) si c est une constante. Noter que $f = 0$ correspond au cas où il n'y a pas de production de polluant pour $t > 0$.

Exercice 1.3

On considère l'équation de convection (1.27) avec la condition initiale (1.28) pour le cas $f(x,t) = 0$ (transport libre).

1. On se propose d'abord de calculer la solution exacte. Pour cela, on introduit le changement de variables

$$X = \alpha x + \beta t, \quad T = \gamma x + \mu t, \qquad (\alpha, \beta, \gamma, \mu \in \mathbb{R}) \tag{1.29}$$

et définit la fonction U par $U(X,T) = u(x,t)$. Ce changement de variables est-il bijectif ? Écrire l'EDP vérifiée par U. Que devient cette équation, si on prend $\beta = -c\alpha$? Résoudre cette dernière équation. En déduire que la solution u est constante sur les droites (C_ξ) du plan (x,t) d'équations

$$x = \xi + ct, \qquad \xi \in \mathbb{R}. \tag{1.30}$$

Définition 1.5 *Les droites (1.30) sont appelées courbes caractéristiques de l'équation de convection (1.27).*

2. On veut maintenant trouver la solution exacte (1.27)-(1.28) sur un intervalle réel $[a,b]$. On suppose que $c > 0$. En traçant dans le plan (x,t) les courbes caractéristiques C_ξ pour $\xi \in [a,b]$, montrer que la solution $u(x,t)$ pour tout $x \in [a,b]$ et tout $t > 0$ est complètement déterminée par la condition initiale u_0 et une condition à la limite

$$u(a,t) = \varphi(t), \qquad \forall t > 0. \tag{1.31}$$

En déduire que pour un T donné la solution exacte est :

$$u(x,t) = \begin{cases} u_0(x - cT) & \text{si } x - cT > a, \\ \varphi(T - \dfrac{x-a}{c}) & \text{si } x - cT < a. \end{cases} \tag{1.32}$$

Remarquer que dans le cas où $u_0(a) \neq \varphi(0)$, la solution est discontinue le long de la caractéristique $x - cT = a$. Déterminer le temps T à partir duquel la condition initiale u_0 est évacuée du domaine $[a, b]$, i.e. $u(x,t)$ s'exprime seulement en fonction de $\varphi(t)$. Indiquer la condition à la limite à imposer pour pouvoir déterminer u dans le cas $c < 0$.

3. Pour résoudre numériquement l'équation de convection (1.27) on définit une discrétisation en espace :

$$x_j = a + j\delta x, \quad \delta x = \frac{b-a}{J}, \quad j = 0, 1, \ldots, J \geqslant 2, \tag{1.33}$$

et une discrétisation en temps : pour $T > 0$ fixé, on définit les instants intermédiaires

$$t_n = n\delta t, \quad \delta t = \frac{T}{N}, \quad n = 0, 1, \ldots, N \geqslant 2. \tag{1.34}$$

Écrire une fonction pour le calcul de la solution exacte (1.32)

```
function uex=conv_exact(a,b,x,T,fun_ci,fun_cl)
% Arguments d'entrée :
%   a,b  l'intervalle de définition [a,b]
%   c>0  la vitesse de convection
%   x    le vecteur x(j) = a + j δ x, j = 0,1,...,J
%   T    l'instant de temps pour lequel on demande la solution
%   fun_ci(x)   la condition initiale en (t = 0)
%   fun_cl(x)   la condition {à} la limite (x = a)
% Argument de sortie :
%   uex  le vecteur de dimension J+1 contenant la solution exacte
```

4. On suppose $c > 0$ et on note u_j^n une approximation de $u(x_j, t_n)$. Considérons l'algorithme numérique suivant pour le calcul de u_j^n :

- Pour $n = 0$ (on impose la condition initiale) : $u_j^0 = u_0(x_j), j = 0, 1, \ldots, J$.
- Pour $n = 0, \ldots, N-1$ (avancement en temps, on calcule u^{n+1})
- Pour $j = 1, \ldots, J$ (intérieur du domaine)

$$u_j^{n+1} = u_j^n - \frac{c\delta t}{\delta x}(u_j^n - u_{j-1}^n). \tag{1.35}$$

- Pour $j = 0$ (condition à la limite) : $u_0^{n+1} = \varphi(t_{n+1})$.

a) Justifier géométriquement (en traçant la caractéristique partant du point u_j^{n+1}) que l'algorithme est bien posé si

$$\sigma = \frac{c\delta t}{\delta x} \leqslant 1. \tag{1.36}$$

Définition 1.6 *La relation (1.36) donne la condition suffisante de stabilité du schéma décentré (1.35) pour l'équation de convection et s'appelle condition CFL (Courant-Friedrichs-Levy).*

b) Écrire un programme utilisant cet algorithme pour résoudre l'équation de convection pour

$$a = 0, \quad b = 1, \quad c = 4, \quad f = 0,$$
$$u_0(x) = x, \quad \varphi(t) = \sin(10\pi t).$$

Prendre $J = 40$ et calculer δt à partir de (1.36) avec $\sigma = 0.8$.

Tracer les solutions obtenues après $n = 10, 20, 30, 40, 50$ pas de temps. Comparer avec la solution exacte et commenter.

Que se passe-t-il si on prend $\sigma = 1$? puis $\sigma = 1.1$? Expliquer.

La solution de cet exercice se trouve à la page 21.

1.2.2 L'équation des ondes

La propagation des ondes (acoustiques, élastiques, sismiques, etc) a comme modèle l'équation aux dérivées partielles (EDP) du second ordre :

$$\partial_{tt}^2 u(x,t) - c^2 \partial_{xx}^2 u(x,t) = 0, \quad t > 0, \tag{1.37}$$

où c est la vitesse de propagation des ondes. Le problème de Cauchy correspondant nécessite deux conditions initiales

$$u(x,0) = u_0(x), \quad \partial_t u(x,0) = u_1(x). \tag{1.38}$$

Exemple 3 : Les oscillations d'une corde élastique sont décrites par l'équation (1.37), où la fonction $u(x,t)$ représente l'amplitude des mouvements transversaux de la corde. La vitesse de propagation c dépend de la tension τ dans la corde et de sa densité linéaire ρ suivant la loi : $c = \sqrt{\tau/\rho}$. Les conditions (1.38) donnent la position et la vitesse initiales de la corde.

Si la corde est considérée infinie, l'équation sera posée sur tout $\mathbb{R}$. Pour une corde de longueur ℓ finie, il faut également imposer des conditions aux limites. Par exemple, si la corde est fixée aux extrémités, les conditions aux limites correspondantes seront :

$$u(0,t) = u(\ell,t) = 0, \quad \forall t > 0. \tag{1.39}$$

Définition 1.7 *Les conditions aux limites (1.39) sont appelées conditions de Dirichlet (on impose les valeurs de la fonction aux bords du domaine de calcul). Quand les valeurs imposées sont nulles, on dit que les conditions aux limites sont homogènes.*

Corde infinie

Dans un premier temps, on considère le cas d'une corde vibrante infinie ($x \in \mathbb{R}$).

Exercice 1.4

Solution exacte pour la corde infinie. En utilisant le changement de variables (1.29), on définit la fonction $U(X,T) = u(x,t)$ et on se propose de calculer la solution exacte.

- Exprimer $\partial^2_{tt}u$ et $\partial^2_{xx}u$ en fonction des dérivées de U. En déduire une EDP vérifiée par U.
- Que devient cette équation, si on choisit $\mu = c\gamma$ et $\beta = -c\alpha$? Montrer qu'il existe une fonction F et une fonction G telles que $U(X,T) = F(X) + G(T)$.
- En déduire que la solution générale de l'équation des ondes s'écrit sous la forme :

$$u(x,t) = f(x-ct) + g(x+ct). \tag{1.40}$$

- En utilisant les conditions initiales (1.38) montrer que

$$u(x,t) = \frac{u_0(x-ct) + u_0(x+ct)}{2} + \frac{1}{2c}\int_{x-ct}^{x+ct} u_1(s)ds. \tag{1.41}$$

La solution de cet exercice se trouve à la page 24.

Domaine de dépendance, condition CFL

L'expression précédente montre que la valeur de u en un point x et à l'instant t ne dépend que des valeurs des données initiales u_0 et u_1 sur l'intervalle $[x-ct, x+ct]$.

Définition 1.8 *Les droites d'équations $x - ct = \xi$ et $x + ct = \xi$, avec $\xi \in \mathbb{R}$ une constante, sont les caractéristiques de l'équation des ondes (1.37).*

Supposons utiliser un schéma numérique pour résoudre l'équation des ondes. À l'instant $t_{n+1} = (n+1)\delta t$, la valeur de la solution u_j^{n+1} au point $x_j = j\delta x$ sera déterminée par l'information transportée du niveau t_n par les deux caractéristiques partant du point (x_j, t_{n+1}) (voir la figure 1.2). Le cône formé par les deux caractéristiques s'appelle domaine de dépendance.

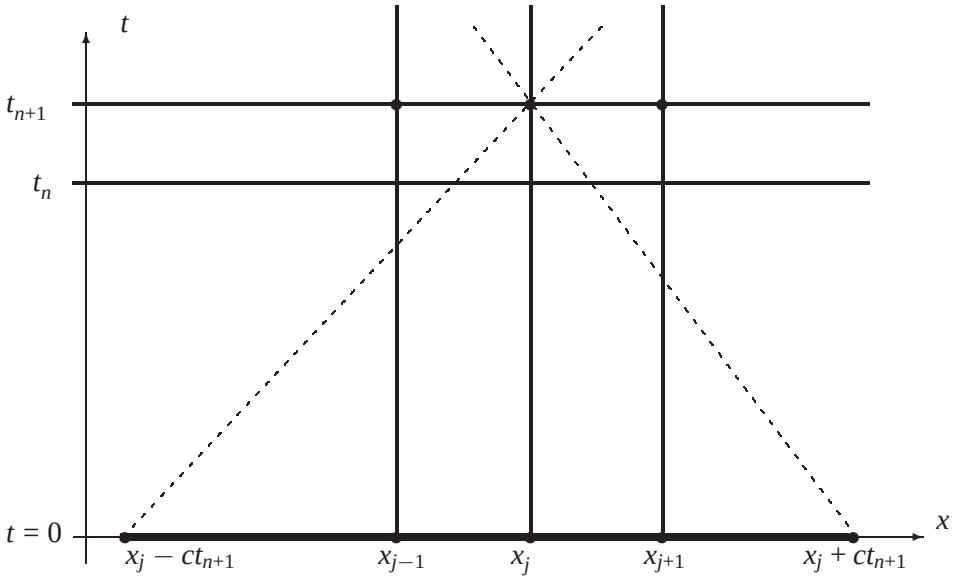


Figure 1.2 Domaine de dépendance pour l'équation des ondes.

Exercice 1.5

Justifier le schéma numérique suivant pour l'équation des ondes :

$$\frac{u_j^{n+1} - 2u_j^n + u_j^{n-1}}{\delta t^2} = c^2 \frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{\delta x^2}. \tag{1.42}$$

Montrer que ce schéma est d'ordre deux en temps et en espace (voir aussi l'équation 1.10) et que la condition de stabilité (ou CFL) est identique à celle trouvée pour l'équation de convection :

$$\sigma = |c| \frac{\delta t}{\delta x} \leqslant 1. \tag{1.43}$$

La solution de cet exercice se trouve à la page 25.

Exercice 1.6

Conditions initiales périodiques. On suppose que les données initiales u_0 et u_1 sont périodiques (de période commune τ). Montrer que la solution $u(x, t)$ de l'équation des ondes est périodique en espace (période τ) et en temps (période τ/c).

1. Justifier l'algorithme suivant :

- Connaissant les conditions initiales $u_0(x_j)$ et $u_1(x_j)$, on calcule u_j^0, u_j^1 par

$$u_j^0 = u_0(x_j), \quad u_j^1 = u_j^0 + \delta t\, u_1(x_j) \tag{1.44}$$

- Pour $n \geqslant 1$, on calcule :

$$u_j^{n+1} = 2(1-\sigma^2)u_j^n + \sigma^2(u_{j-1}^n + u_{j+1}^n) - u_j^{n-1}. \tag{1.45}$$

2. Écrire un programme réalisant cet algorithme.
3. Exécuter le programme pour une longueur de corde $\ell = 1$ et une vitesse des ondes $c = 2$. Les données initiales sont $u_0(x) = \sin(2\pi x) + sin(10\pi x)/4$ et $u_1(x) = 0$, ce qui correspond à une corde initialement au repos. Quelle est la périodicité en temps de la solution ?
4. Pour $nx = 50$ points de discrétisation en espace et $nt = 50$ points de discrétisation pour une période de temps, représenter la solution exacte et les solutions obtenues après une et deux périodes. Vérifier que le schéma préserve la périodicité de la solution. Même question pour $nx = 51$. Commenter.

La solution de cet exercice se trouve à la page 25.

Corde vibrante finie

Considérons l'équation des ondes (1.37) avec les conditions initiales (1.38) et les conditions aux limites (1.39). La solution est cherchée sous la forme suivante (décomposition en ondes simples ou de Fourier) :

$$u(x,t) = \sum_{k\in\mathbb{N}^*} \hat{u}_k(t)\phi_k(x), \quad \phi_k(x) = \sin(\frac{k\pi}{\ell}x). \tag{1.46}$$

Pour chaque onde ϕ_k, on appelle k le nombre d'onde et $\hat{u}_k$ l'amplitude de l'onde.

Exercice 1.7

1. Écrire et résoudre l'EDO vérifiée par chaque fonction $\hat{u}_k$.
2. Montrer que la solution exacte pour la corde vibrante finie est :

$$u(x,t) = \sum_{k\in\mathbb{N}^*} [A_k \cos(\frac{k\pi}{\ell}ct) + B_k \sin(\frac{k\pi}{\ell}ct)]\phi_k(x), \tag{1.47}$$

avec

$$A_k = \frac{2}{\ell}\int_0^\ell u_0(x)\phi_k(x)dx, \qquad B_k = \frac{2}{k\pi c}\int_0^\ell u_1(x)\phi_k(x)dx. \tag{1.48}$$

En déduire la période en temps et en espace de la solution.

3. Écrire un programme pour résoudre le problème de la corde vibrante finie en utilisant le schéma centré (1.42). On utilisera le programme développé à l'exercice précédent, en prenant garde d'utiliser les nouvelles conditions aux limites !

Trouver la solution exacte correspondant aux conditions initiales suivantes :

$$u_0(x) = \sin(\frac{\pi}{\ell}x) + \frac{1}{4}\sin(10\frac{\pi}{\ell}x), \quad u_1(x) = 0. \tag{1.49}$$

Tracer la solution exacte et la solution numérique pour plusieurs instants de temps sur une période. Données numériques : $c = 2$, $\ell = 1$, $nx = 50$, $nt = 125$.

La solution de cet exercice se trouve à la page 26.

1.2.3 L'équation de la chaleur

Les phénomènes de diffusion (moléculaire, de la chaleur, etc.) ont comme modèle de description mathématique l'équation de la chaleur :

$$\partial_t u - \kappa\partial^2_{xx}u = f(x,t), \quad \forall t > 0, \tag{1.50}$$

avec la condition initiale :

$$u(x,0) = u_0(x). \tag{1.51}$$

Exemple 4 : La température θ d'un corps chauffé est solution de l'équation

$$\partial_t\theta - \partial_x(\kappa\partial_x\theta) = f(x,t), \tag{1.52}$$

où κ est la diffusivité thermique du matériau et f modélise la source de chaleur. Dans un matériau homogène, κ ne dépend pas de la position x et on retrouve l'équation modèle (1.50).

Considérons le problème d'un mur d'épaisseur ℓ, qui se trouve initialement à une température uniforme θ_0 (température de la chambre). À l'instant $t = 0$, la température extérieure (en $x = 0$) monte brusquement à $\theta_s > \theta_0$, valeur maintenue constante par une source de chaleur. On suppose que la température à $x = \ell$ est gardée à sa valeur initiale θ_0. La propagation de la chaleur dans le mur sera décrite par l'équation de la chaleur (1.50), avec l'inconnue $u(x,t) = \theta(x,t) - \theta_0$, $f(x,t) = 0$, la condition initiale $u_0(x) = 0$ et les conditions aux limites de Dirichlet :

$$u(0,t) = \theta_s - \theta_0 = u_s, \quad u(\ell,t) = 0, \quad \forall t > 0. \tag{1.53}$$

Cas du domaine infini

Pour un mur d'épaisseur infinie ($\ell \to \infty$) on va chercher la solution sous la forme :

$$u(x,t) = f(\eta), \quad \text{avec} \quad \eta = \frac{x}{2\sqrt{\kappa t}}. \tag{1.54}$$

Exercice 1.8

Montrer que la fonction f vérifie l'EDO suivante :

$$\frac{d^2f}{d\eta^2} + 2\eta\frac{df}{d\eta} = 0. \tag{1.55}$$

En introduisant la fonction suivante, appelée *fonction erreur*

$$\text{erf}(z) = \frac{2}{\sqrt{\pi}} \int_0^z e^{-\zeta^2} d\zeta, \tag{1.56}$$

qui vérifie erf(0) = 0 et erf(∞) = 1, trouver la solution de l'équation de la chaleur pour $\ell \to \infty$:

$$u(x,t) = [1 - \text{erf}(\frac{x}{2\sqrt{\kappa t}})]u(0,t). \tag{1.57}$$

La solution de cet exercice se trouve à la page 27.

Remarque 1.2 : Le changement de la valeur de la solution au point $x = 0$ a eu comme conséquence la modification de la solution en tout point du domaine. Le domaine de dépendance pour l'équation de la chaleur est, par conséquent, le domaine de définition tout entier. De façon équivalente, on peut dire que la perturbation introduite en $x = 0$ s'est propagée instantanément dans le domaine de calcul ($u(x,t) > 0, \forall x$ dans la formule 1.57). On dit que la vitesse de propagation est infinie. Rappelons que pour l'équation des ondes le domaine de dépendance était la zone délimitée par les caractéristiques et que la vitesse de propagation de l'information était finie.

Cas du domaine fini

Pour un mur d'épaisseur finie ℓ, on va utiliser la décomposition en ondes simples (1.46).

Exercice 1.9

Écrire et résoudre l'EDO vérifiée par chaque fonction $\hat{u}_k$. Vérifier que la solution de l'équation de la chaleur avec les conditions aux limites (1.53) s'écrit sous la forme :

$$u(x,t) = (1 - \frac{x}{\ell})u_s + \sum_{k \in \mathbb{N}^*} A_k \exp\left(-\left(\frac{k\pi}{\ell}\right)^2 t\right) \phi_k(x). \tag{1.58}$$

Montrer que $A_k = -\dfrac{2u_s}{k\pi}$.

La solution de cet exercice se trouve à la page 28.

Remarque 1.3 : Comparons la solution analytique (1.58) avec celle obtenue pour l'équation des ondes (1.47). L'équation des ondes décrit un phénomène de transport en temps de la condition initiale (l'amplitude de chaque onde ϕ_k reste constante). Le phénomène de diffusion décrit par l'équation de la chaleur se manifeste par la décroissance (rapide) en temps de l'amplitude de chaque onde élémentaire ϕ_k (présence du facteur exponentiel). Cet effet de *lissage* de l'opérateur de la chaleur est d'autant plus important que le nombre d'onde k est grand.

Exercice 1.10

Résolution numérique. Considérons le schéma explicite centré suivant pour l'équation de la chaleur :

$$\frac{u_j^{n+1} - u_j^n}{\delta t} - \kappa \frac{u_{j+1}^n - 2u_j^n + u_{j-1}^n}{\delta x^2} = 0. \tag{1.59}$$

La condition de stabilité du schéma est :

$$\kappa \frac{\delta t}{\delta x^2} \leqslant \frac{1}{2} \tag{1.60}$$

1. Écrire un programme pour la résolution du problème de la propagation de la chaleur dans un mur de'épaisseur finie. On prendra $\kappa = 1$, $\ell = 1$, $u_s = 1$, $nx = 50$ points de discrétisation en espace et le pas de temps δt donné par (1.60). Tracer la solution numérique pour différents instants de temps et comparer avec la solution exacte (1.58) (une bonne approximation de cette dernière est obtenue en prenant les 20 premiers nombres d'onde k). Comparer également avec la solution (1.57) obtenue pour un domaine infini. Commenter les résultats pour t petit et pour t grand.
2. *Effet lissant.* Reprendre le programme précédent pour $u_s = 0$ et la condition initiale

$$u_0(x) = u(x, 0) = \sin(\frac{\pi}{\ell}x) + \frac{1}{4}\sin(10\frac{\pi}{\ell}x). \tag{1.61}$$

Comparer la solution numérique avec la solution analytique donnée par (1.58). Commenter par rapport au résultat obtenu pour l'équation des ondes avec la même condition initiale. Décrire l'amortissement des ondes présentes dans la condition initiale.

La solution de cet exercice se trouve à la page 28.

1.3 EN SAVOIR PLUS

La résolution numérique des équations aux dérivées partielles (EDP) ou des équations différentielles (EDO) ou est un thème majeur de l'analyse numérique. Il existe une nombreuse littérature sur l'étude de la stabilité et la convergence des schémas numériques. Le lecteur pourra consulter
[Crouzeix et Mignot, 1989], [Delabrière et Postel, 2004] et [Demailly, 1996] pour l'analyse numérique des EDO. Pour les EDP, nous renvoyons à [Hirsh, 1988], [Lucquin et Pironneau, 1996] et [Mohammadi et Saïac, 2003]. L'implémentation des méthodes de résolution en utilisant des langages de programmation évolués est présentée dans [Danaila, Hecht et Pironneau, 2003].

1.4 SOLUTIONS ET PROGRAMMES

Solution des exercices 1.1 et 1.2. (L'équation d'absorbtion)

1. En remplaçant $u(t) = e^{-\alpha t}v(t)$ dans (1.22), on obtient que $v(t)$ est solution de l'équation différentielle $v'(t) = e^{\alpha t}f(t)$ avec la condition initiale $v(0) = u(0) = u_0$; par intégration, le résultat (1.23) est immédiat.
2. On obtient :
$$u(t) = e^{-\int_0^t \alpha(s)ds}\left[u_0 + \int_0^t e^{-\int_0^z \alpha(s)ds} f(z)dz\right].$$
3. On suppose la fonction $f(t) = f$ constante. L'expression (1.23) devient :
$$u(t) = \frac{f}{\alpha} + e^{-\alpha t}(u_0 - \frac{f}{\alpha}).$$
Si $u_0 = f/\alpha$, la solution est constante et pour tout t, $u(t) = u_0$.
Pour $\alpha > 0$, $u(t) \to f/\alpha$ quand $t \to \infty$.
Pour $\alpha < 0$, $u(t) \to +\infty \times$ signe de $(u_0 - f/\alpha)$.
4. Si $\alpha = \alpha_r + \mathbf{i}\alpha_i$, on obtient
$$|u(t)| = |e^{-\alpha t}u_0| = |e^{-(\alpha_r + \mathbf{i}\alpha_i)t}u_0| = |e^{-\alpha_r t}u_0| \to 0.$$
5. Les fonctions MATLAB *EulerExp.m* et *RKutta4.m* utilisent, respectivement, le schéma d'Euler explicite et le schéma de Runge-Kutta d'ordre 4. pour l'intégration de l'EDO $u'(t) = f(t, u)$. Le second membre $f(t, u)$ porte le nom générique `fun` à l'intérieur des fonctions ; il sera effectivement précisé et évalué au moment de l'appel des fonctions (voir plus bas). Les deux fonctions retournent un vecteur contenant les valeurs u_k de la solution numérique, calculés pour des instants de temps distribués uniformément entre t_0 et t_1.

Le programme MATLAB *Eq_absorb.m* appelle les fonctions `EulerExp` et `RKutta4` en utilisant comme argument d'entrée la fonction `fabsorb` (écrite dans un fichier séparé), décrivant l'équation d'absorbtion. Les résultats sont représentés graphiquement sur deux figures distinctes. Le programme trace également les domaines de stabilité des schémas numériques (exercice 1.2).

Les résultats sont groupés sur la figure 1.3. Observons que pour $h = 1/8 < 2/\alpha = 1/2$ tout se passe bien : la solution numérique approche la solution exacte, avec une meilleure approximation pour le schéma de Runge-Kutta (superposition des deux courbes). Par contre, pour $h = 1/2 = 2/\alpha$ on atteint la limite de stabilité du schéma d'Euler explicite. La suite des valeurs numériques ne converge pas, même si elle reste bornée (ce qui n'est plus le cas pour $h > 1/2$ - à tester).

Le schéma de Runge-Kutta d'ordre 4, qui a un domaine de stabilité plus étendu, reste stable pour les deux valeurs considérées du pas de discrétisation h. La figure 1.4 montre que le domaine de stabilité du schéma *RKutta*4 inclut les domaines des schémas Euler explicite et *RKutta*2.

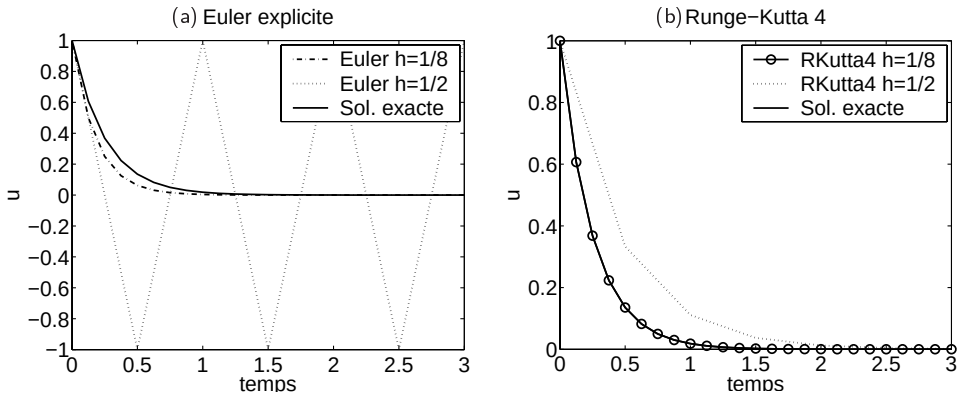


Figure 1.3 Calcul par un schéma d'Euler explicite (a) et un schéma de Runge-Kutta d'ordre 4 (b) de la solution de l'EDO : $u'(t) + 4u(t) = 0$. La solution exacte est représentée en trait continu.

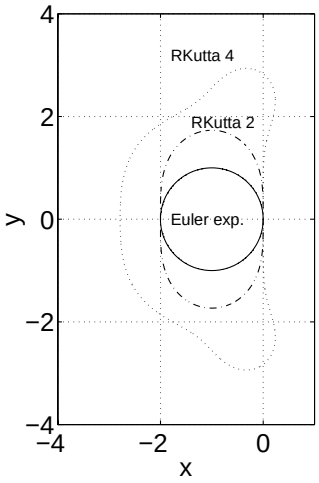


Figure 1.4 Domaine de stabilité pour les schémas numériques.

Au vu de ces résultats, le schéma *RKutta*4 semble être le meilleur choix, en offrant la meilleure stabilité et précision. En pratique, le choix d'un schéma ou d'un autre est dicté par le compromis entre ses caractéristiques (précision, stabilité) et le coût de calcul engendré (le schéma *RKutta*4 est approximativement 4 fois plus coûteux que le schéma d'Euler explicite pour le même pas de temps).

Solution de l'exercice 1.3. (L'équation de convection)

1. Le changement de variable s'écrit sous la forme

$$\begin{pmatrix} X \\ T \end{pmatrix} = \begin{pmatrix} \alpha & \beta \\ \gamma & \mu \end{pmatrix} \begin{pmatrix} x \\ t \end{pmatrix}$$

et il est bijectif si $\alpha\mu \neq \beta\gamma$. La dérivation par rapport aux nouvelles variables donne :

$$\partial_t u = \beta \partial_X U + \mu \partial_T U, \quad \partial_x u = \alpha \partial_X U + \gamma \partial_T U, \tag{1.62}$$

et donc U est solution de l'EDP

$$(\beta + c\alpha)\, \partial_X U + (\mu + c\gamma)\, \partial_T U = 0.$$

Pour $\beta = -c\alpha$, on obtient $(\mu + c\gamma)\, \partial_T U = 0$, soit $\partial_T U = 0$. Cette dernière équation a pour solution $U(X, T) = F(X)$, où F est n'importe quelle fonction. On a donc $u(x, t) = F(X) = F(\alpha x + \beta t) = F(\alpha x - \alpha c t) = G(x - ct)$ où G est une fonction quelconque. En imposant la condition initiale (1.28) nous obtenons $u(x, t) = u_0(x - ct)$. On en déduit en particulier que la solution reste constante le long des droites caractéristiques

$$\text{si} \quad (x, t) \in C_\xi \Longrightarrow u(x, t) = u_0(\xi).$$

Dans le plan (x, t), les courbes caractéristiques C_ξ sont des droites de pentes positives, égales à $1/c$ (voir la figure 1.5).

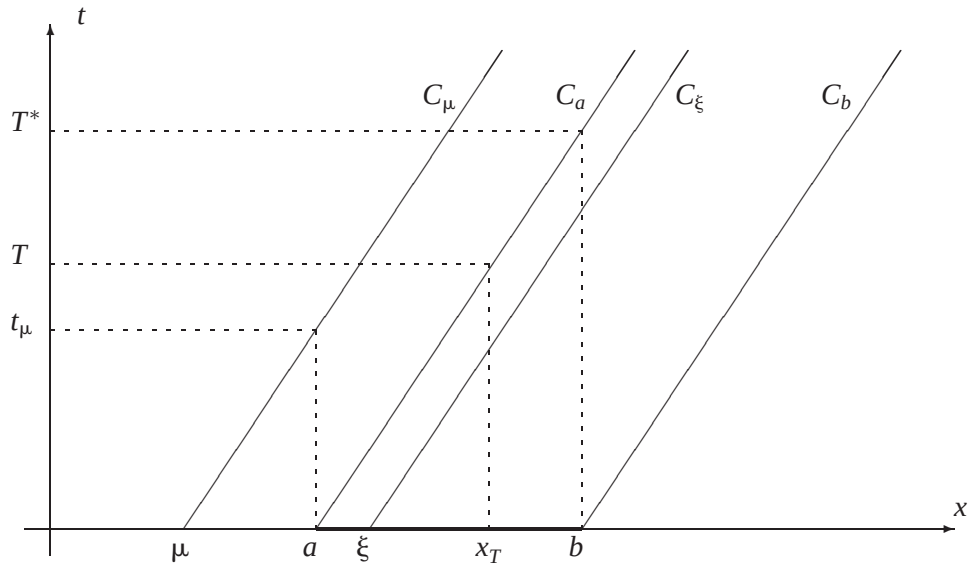


Figure 1.5 Calcul de la solution exacte de l'équation de convection par la méthode des caractéristiques.

2. Pour déterminer la solution $u(x,T)$ pour $x \in [a,b]$ et $T < T^* = (b-a)/c$, il suffit de tracer les caractéristiques qui passent par les points (x,T) et d'utiliser le fait que $u(x,t)$ est constant le long d'une caractéristique. Deux cas sont possibles :

- la caractéristique (C_ξ dans le dessin) coupe le segment $[a,b]$; c'est le cas des points $x \geqslant x_T$, avec $x_T = a + cT$. La solution sera donc donnée par la condition initiale :
$$u(x,T) = u_0(\xi) = u_0(x - cT).$$
- la caractéristique (C_μ dans le dessin) ne coupe pas le segment $[a,b]$; dans ce cas il faut imposer une condition à la limite $u(a,t) = \varphi(t)$. L'information sera propagée à partir de cette donnée :
$$u(x,T) = \varphi(t_\mu) = \varphi(T - \frac{x-a}{c}).$$

Observons que la donnée initiale u_0 est complètement évacuée du domaine de définition $[a,b]$ après le temps $T^* = (b-a)/c$.

Pour $c < 0$, c'est à droite qu'il faut imposer une condition à la limite sous la forme $u(b,t) = \varphi(t)$.

3. La fonction MATLAB *conv_exact.m* calcule la solution exacte pour un temps T donné. Remarquons l'utilisation de la commande MATLAB `find` pour implémenter la formule (1.32).

4. On reconnaît une discrétisation de l'équation (1.27) où la dérivée en temps $\partial_t u$ est approchée par $D^+u(t_n)$ et la dérivée en espace $\partial_x u$ est approchée par $D^-u(x_j)$. Le choix de l'approximation décentrée à gauche pour $\partial_x u$ est lié au fait que $c > 0$, donc l'information vient de la gauche.

La caractéristique passant par le point u_j^{n+1} est représentée sur la figure 1.6. Cette droite coupe la droite $t = t_n$ en un point P compris entre x_{j-1} et x_j, situé à la distance $c\delta t$ de x_j et $\delta x - c\delta t$ de x_{j-1}. Comme la solution est conservée le long de la caractéristique, nécessairement $u_j^{n+1} = u_P^n$. Observons que le schéma (1.35) calcule la valeur u_P^n par interpolation linéaire entre les valeurs prises par la solution aux points x_{j-1} et x_j :

$$u_j^{n+1} = \frac{\delta x - c\delta t}{\delta x} u_j^n + \frac{c\delta t}{\delta x} u_{j-1}^n.$$

La condition CFL : $c\delta t \leqslant \delta x$, exprime le fait que les coefficients de cette interpolation doivent être positifs, autrement dit, le point P doit se trouver à l'intérieur de l'intervalle $[x_{j-1}, x_j]$.

Pour la programmation, on fera attention à faire démarrer les indices des tableaux à partir de 1 (et non pas de 0). La solution à l'instant t_{n+1} sera calculée par la relation :

$$u_j^{n+1} = (1-\sigma)u_j^n + \sigma u_{j-1}^n, \quad \sigma = \frac{c\delta t}{\delta x}.$$

Pour réduire le stockage des variables, il est possible d'utiliser un seul tableau u pour la solution (les valeurs de ce tableau seront stockées dans un fichier ou représentées graphiquement pour des instants de temps choisis). Par conséquent, la relation

précédente devient :

$$u(j) = (1 - \sigma)u(j) + \sigma u(j - 1),$$

et les valeurs calculées à l'instant t_{n+1} seront stockées en écrasant les valeurs du niveau t_n. Afin de s'assurer que les valeurs u_{j-1} à l'instant précédent t_n sont utilisées, les indices j seront balayés de $J + 1$ à 2 (boucle inverse). Pour $j = 1$ la condition à la limite sera imposée.

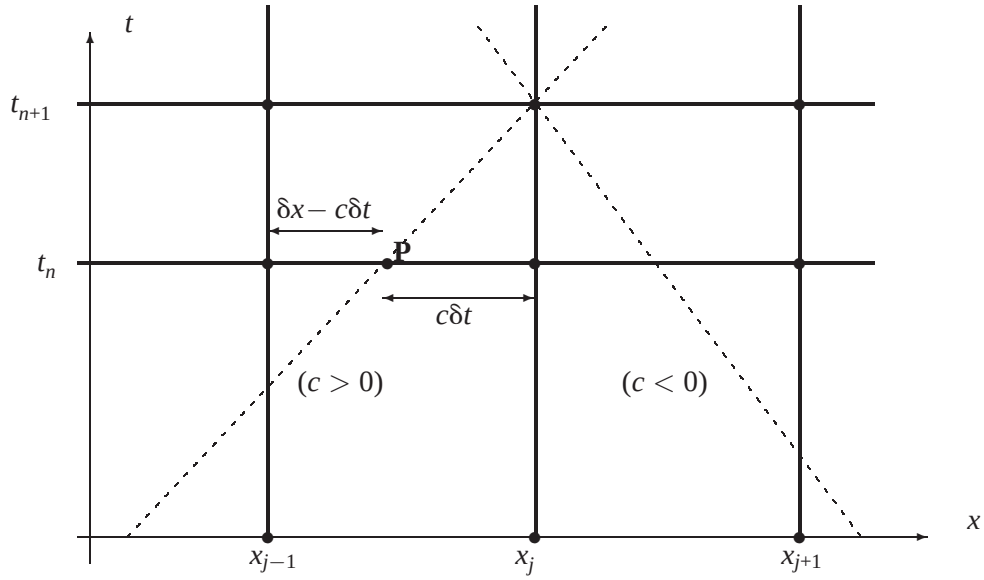


Figure 1.6 Interprétation géométrique de la condition CFL pour le schéma décentré.

Ce type de calcul est implémenté dans le programme MATLAB *Eq_convec.m*. Ce programme appelle les fonctions `conv_ci` et `conv_cl` (fichiers séparés) qui définissent, respectivement, la condition initiale et la condition à la limite.

La solution pour les instants demandés est représentée sur la figure 1.7 et comparée avec la solution exacte (appel de la fonction `conv_exact`). On note qu'à partir de l'instant $T^* = 1/4$, la donnée initiale sort du domaine de calcul et la solution ne dépend plus que de la condition à la limite.

On observe un effet numérique de *lissage* des zones où la dérivée de la solution exacte est discontinue. Il s'agit d'un effet dissipatif qui sera analysé dans les paragraphes suivants (équation de la chaleur). Ce qui est intéressant dans notre cas est le fait que la dissipation observée n'a rien de physique - elle est introduite (et en même temps cachée) par le schéma numérique.

Un calcul avec $\sigma = 1$ (ou CFL=1) montre une superposition parfaite entre la solution exacte et celle numérique. Rien d'étonnant, car le schéma décentré devient une relation exacte : $u_j^{n+1} = u_{j-1}^n$. En pratique, la vitesse de convection c et le pas d'espace δx sont généralement variables, ce qui rend impossible un calcul avec $\sigma = 1$ partout.

Le calcul avec $\sigma = 1.1$ prouve la perte de stabilité du schéma décentré.

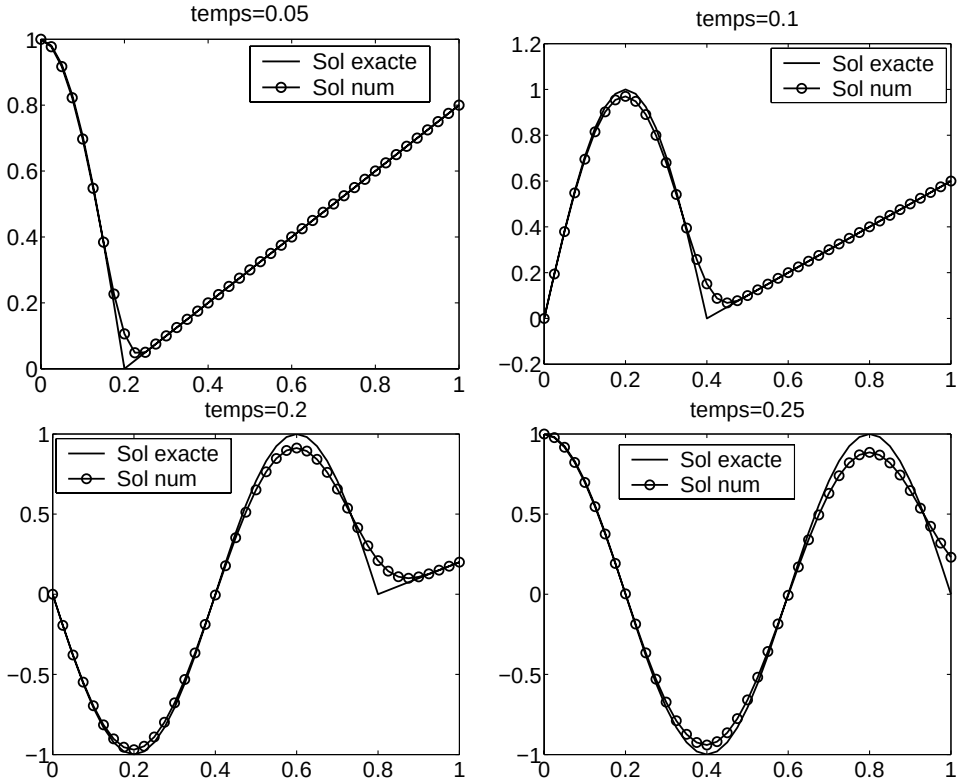


Figure 1.7 Calcul de la solution de l'équation de convection par un schéma décentré (CFL=0.8). La solution exacte est représentée en trait continu.

Solution de l'exercice 1.4. (L'équation des ondes)

À partir de (1.62), nous obtenons pour les dérivées secondes :

$$\begin{cases} \partial^2_{tt} u &= \beta^2 \partial^2_{XX} U + \mu^2 \partial^2_{TT} U + 2\beta\mu \partial^2_{TX} U \\ \partial^2_{xx} u &= \alpha^2 \partial^2_{XX} U + \gamma^2 \partial^2_{TT} U + 2\alpha\gamma \partial^2_{TX} U \end{cases}$$

et donc U est solution de l'EDP

$$\left(\mu^2 - c^2\gamma^2\right) \partial^2_{TT} U + \left(\beta^2 - c^2\alpha^2\right) \partial^2_{XX} U + 2\left(\beta\mu - c^2\alpha\gamma\right) \partial^2_{XT} U = 0.$$

Pour $\mu = c\gamma$ et $\beta = -c\alpha$, l'équation devient $-4c^2\alpha\gamma\partial^2_{XT} U = 0$, soit $\partial^2_{XT} U = 0$, ou encore $\partial_T(\partial_X U) = 0$. On en déduit qu'il existe une fonction F telle que $\partial_X U = F(X)$, puis qu'il existe une fonction G telle que $U(X, T) = F(X)+G(T)$. Comme $X = \alpha(x-ct)$ et $T = \gamma(x + ct)$, on peut prendre $\alpha = \gamma = 1$ et la solution devient :

$$u(x, t) = f(x - ct) + g(x + ct).$$

En imposant les conditions initiales dans les égalités $u(x,t) = f(x-ct) + g(x+ct)$ et $\partial_t u(x,t) = -cf'(x-ct) + cg'(x+ct)$, on obtient

$$\begin{cases} f'(x) + g'(x) &= u_0'(x) \\ -f'(x) + g'(x) &= (1/c)u_1(x) \end{cases}$$

d'où les expressions de f' et g' et finalement la formule (1.41) pour u.

Solution de l'exercice 1.5

On reconnaît une discrétisation de l'équation des ondes où les dérivées secondes sont approchées par des différences finies centrées (cf. équation 1.10). Par conséquent, le schéma (1.42) est d'ordre deux en temps et en espace.

La condition de stabilité (1.43) exprime le fait que le domaine de dépendance (voir la figure 1.2) délimité par les deux caractéristiques partant du point (x_j, t_{n+1}) doit être contenu dans le domaine *numérique* imposé par le schéma, qui est, dans ce cas, le triangle $\{(x_j, t_{n+1}), (x_{j-1}, t_n), (x_{j+1}, t_n)\}$. Autrement dit, si le pas de temps dépasse la valeur critique $\delta x/c$, l'information sera cherchée par les caractéristiques en dehors du l'intervalle $[x_{j-1}, x_{j+1}]$ considéré par le schéma, ce qui n'est pas cohérent avec le phénomène physique décrit par l'équation des ondes.

D'après (1.41), il est facile de vérifier que :

$$u(x+\tau, t) = u(x,t) \quad \text{et que} \quad u(x, t+\tau/c) = u(x,t).$$

La solution $u(x,t)$ est donc périodique en temps et en espace, de période τ en espace et τ/c en temps.

Solution de l'exercice 1.6

Dans l'algorithme proposé, on utilise le schéma (1.42) avec le calcul de la solution au premier pas de temps basé sur l'approximation $\partial_t u(x,t) \simeq u_1(x)$, valide pour t petit.

Le programme MATLAB mettant en œuvre cet algorithme est *Eq_corde_inf.m*. Faisons quelques commentaires sur la programmation. La condition de périodicité (consistante avec la condition initiale) s'exprime dans notre cas sous la forme $u_{nx+1} = u_1$, où la discrétisation spatiale est faite de telle manière que $x_1 = 0$ et $x_{nx+1} = 1$. Afin d'utiliser pleinement les capacités de calcul vectoriel de MATLAB, nous avons défini les vecteurs *jp* et *jm* correspondant à tous les indices $j+1$, respectivement $j-1$ (en tenant compte, bien entendu, de la condition de périodicité). Le schéma (1.45) se traduit en langage MATLAB par une seule ligne :

```
u2=-u0+coeff*u1+sigma2*(u1(jm)+u1(jp)),
```

où $u2$ correspond au vecteur $(u^{n+1})_j$, $u1$ à $(u^n)_j$ et $u0$ à $(u^{n-1})_j$. L'avantage de ce type de programmation compacte est d'éviter une boucle suivant l'indice j et le traitement particulier au voisinage des bords du domaine de calcul. Ce type de programmation sera utilisé également dans le projet 12.

Les résultats sont montrés sur la figure 1.8. La période en temps de la solution est $1/c = 0.5$. Pour $nx = nt = 50$, on obtient le nombre CFL $\sigma = 1$. Le schéma numérique propage la condition initiale correctement et garde la périodicité en temps (superposition avec la solution exacte qui est en fait la condition initiale u_0). Contrairement aux schémas décentrés utilisés pour l'équation de convection, aucune diffusion numérique n'est observée pour ce schéma centré (diminuer nx pour avoir des $CFL < 1$). Pour $nx = 51$ le schéma devient instable car $CFL > 1$. Il est intéressant d'observer que l'instabilité du schéma se manifeste après un certain temps (ici après la première période).

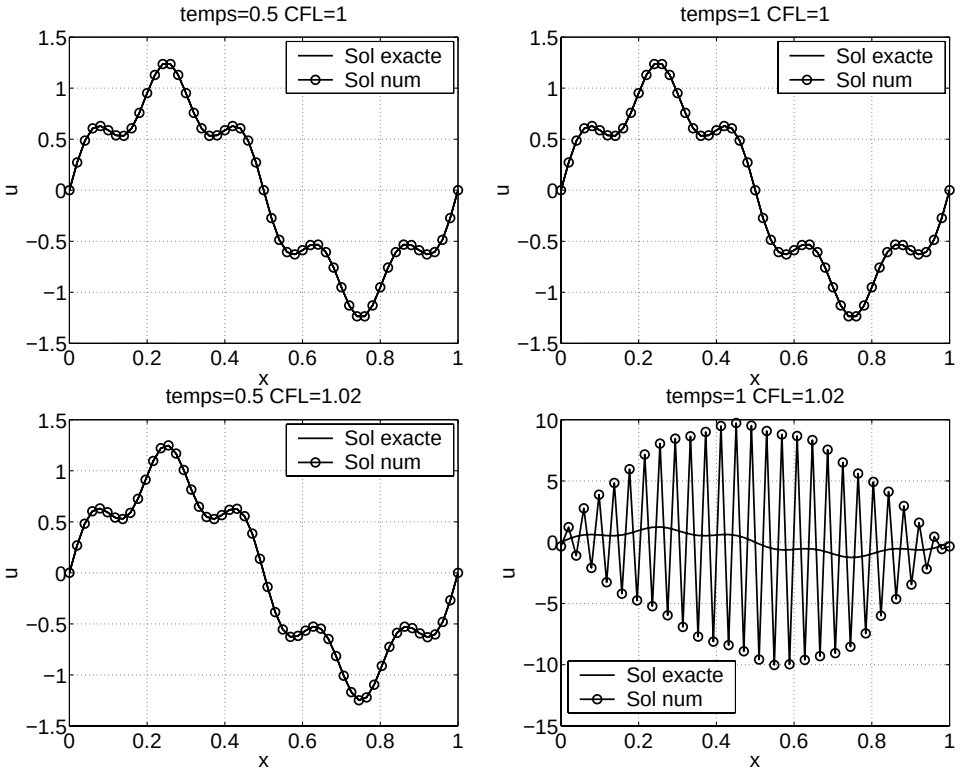


Figure 1.8 Solution numérique de l'équation des ondes pour la corde vibrante infinie (conditions de périodicité). Condition initiale : $u(x,0) = \sin(2\pi x) + \sin(10\pi x)/4$ et $\partial u/\partial t(x,0) = 0$. Comparaison avec la solution exacte pour $CFL = 1$ (en haut) et $CFL > 1$ (en bas) après une période temporelle (à gauche) et deux périodes (à droite).

Solution de l'exercice 1.7

Les amplitudes $\hat{u}_k$ vérifient l'EDO souvent rencontrée en physique (équation du pendule oscillant) :

$$\frac{d^2}{dt^2}\hat{u}_k + c^2\left(\frac{k\pi}{\ell}\right)^2 \hat{u}_k = 0. \tag{1.63}$$

La solution générale de cette EDO étant

$$\hat{u}_k(t) = A_k \cos\left(\frac{k\pi}{\ell}ct\right) + B_k \sin\left(\frac{k\pi}{\ell}ct\right), \tag{1.64}$$

l'expression (1.47) est immédiate. Les coefficients A_k, B_k sont calculés en utilisant l'orthogonalité des fonctions ϕ_k sur $[0, \ell]$.

On observe que la solution est périodique, de période 2ℓ en espace et $2\ell/c$ en temps.

La condition initiale (1.49) correspond à une décomposition en ondes simples avec les valeurs

$$k = \{1, 10\}, \quad A_k = \{1, 1/4\}, \quad B_k = \{0, 0\}.$$

La solution exacte sera donnée par (1.47) avec ces valeurs.

Le programme MATLAB *Eq_corde_finie.m* calcule la solution pour la corde vibrante finie. La condition initiale est calculée dans *ondes_cvf0.m* et la solution exacte dans *ondes_cvfex.m*.

Remarquons à nouveau l'écriture vectorielle (sans utiliser de boucles) du schéma centré. Les points de calcul correspondant aux bords ne sont pas réactualisés en temps, en gardant ainsi les valeurs imposées initialement (respectant les conditions aux limites). La figure 1.9 montre la comparaison entre la solution exacte et la solution numérique pour deux instants de temps.

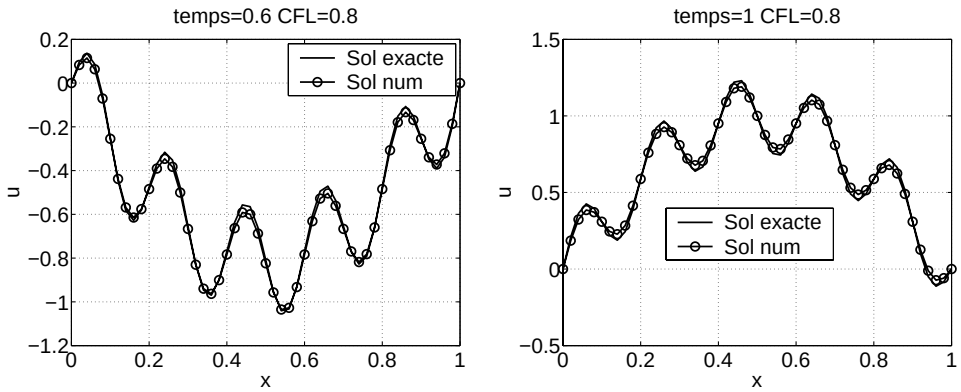


Figure 1.9 Solution numérique de l'équation des ondes pour la corde vibrante finie. Condition initiale : $u(x, 0) = \sin(\pi x) + \sin(10\pi x)/4$ et $\partial u/\partial t(x, 0) = 0$. Comparaison avec la solution exacte (une période correspond à $t = 1$).

Solution de l'exercice 1.8. (L'équation de la chaleur)

Les dérivées partielles s'expriment en fonction de f par :

$$\frac{\partial u}{\partial t} = -\frac{\eta}{2t}\frac{df}{d\eta}, \quad \frac{\partial^2 u}{\partial x^2} = \frac{1}{4kt}\frac{d^2 f}{d\eta^2},$$

d'où l'EDO (1.55) pour f. Par intégration, on obtient :

$$\frac{df}{d\eta} = Ae^{-\eta^2} \Longrightarrow u(x,t) = f(\eta) = B + A\,\mathrm{erf}(\eta).$$

En tenant compte des propriétés de la fonction erf pour imposer les conditions aux limites, on retrouve facilement la formule (1.57).

Solution de l'exercice 1.9

L'EDO vérifiée par les amplitudes $\hat{u}_k$ est :

$$\frac{d\hat{u}_k}{dt} + \left(\frac{k\pi}{\ell}\right)^2 \hat{u}_k = 0,$$

avec la solution

$$\hat{u}_k(t) = A_k \exp\left[-\left(\frac{k\pi}{\ell}\right)^2 t\right].$$

Chaque onde simple $\hat{u}_k(t)\phi_k(x)$ vérifie l'équation de la chaleur, mais pas les conditions aux limites (1.53). C'est la raison pour laquelle une fonction linéaire en x (qui est également solution de l'équation de la chaleur) a été ajoutée à la forme finale de la solution (1.58). Cette superposition de plusieurs solutions est rendue possible par le caractère linéaire de l'équation de la chaleur.

Pour déterminer les coefficients A_k on utilise l'orthogonalité des fonctions ϕ_k et on obtient :

$$A_k = -\frac{2u_s}{\ell}\int_0^l \left(1 - \frac{x}{\ell}\right)\sin\left(\frac{k\pi}{\ell}x\right)dx = -\frac{2u_s}{k\pi}.$$

Solution de l'exercice 1.10

Le programme MATLAB *Eq_chaleur.m* répond aux questions 1 et 2. La condition initiale est calculée dans *chaleur_u0.m* et la solution exacte dans *chaleur_uex.m* (la fonction `erf` est déjà programmée dans les bibliothèques MATLAB).

Les résultats (figure 1.10) montrent que la solution erf (1.57) obtenue pour un domaine infini est une bonne approximation pour des temps t petits (c'est la raison pour laquelle elle est souvent utilisée par les ingénieurs). Pour des temps t plus grands, la solution exacte (et celle numérique heureusement) converge vers la solution "stationnaire" (qui ne dépend plus du temps) qui est la fonction linéaire $u(x) = \left(1 - \frac{x}{\ell}\right) u_s$.

> **Remarque 1.4 :** Le phénomène de diffusion décrit par l'équation de la chaleur est caractérisé par une échelle de temps $t_0 = \ell^2/\kappa$ (voir l'expression de η dans 1.54). Par conséquent, la vitesse effective de propagation de la perturbation thermique $c_0 = \ell/t_0 = \kappa/\ell$, décroît avec la distance, ce qui explique l'inefficacité des mécanismes de diffusion à grande distance (ou pour des temps longs) ! Prenons l'exemple du chauffage d'une habitation : la diffusivité thermique de l'air étant $\kappa \approx 20$ [mm^2/s], l'effet du chauffage sera ressenti à 1 cm au bout

de 5 secondes et à 1 mètre après $5{\cdot}10^4$ s $\approx$ 14 heures ! Heureusement, les phénomènes de convection y intervenant sont caractérisés par une vitesse de propagation constante, rendant le processus de chauffage plus efficace !

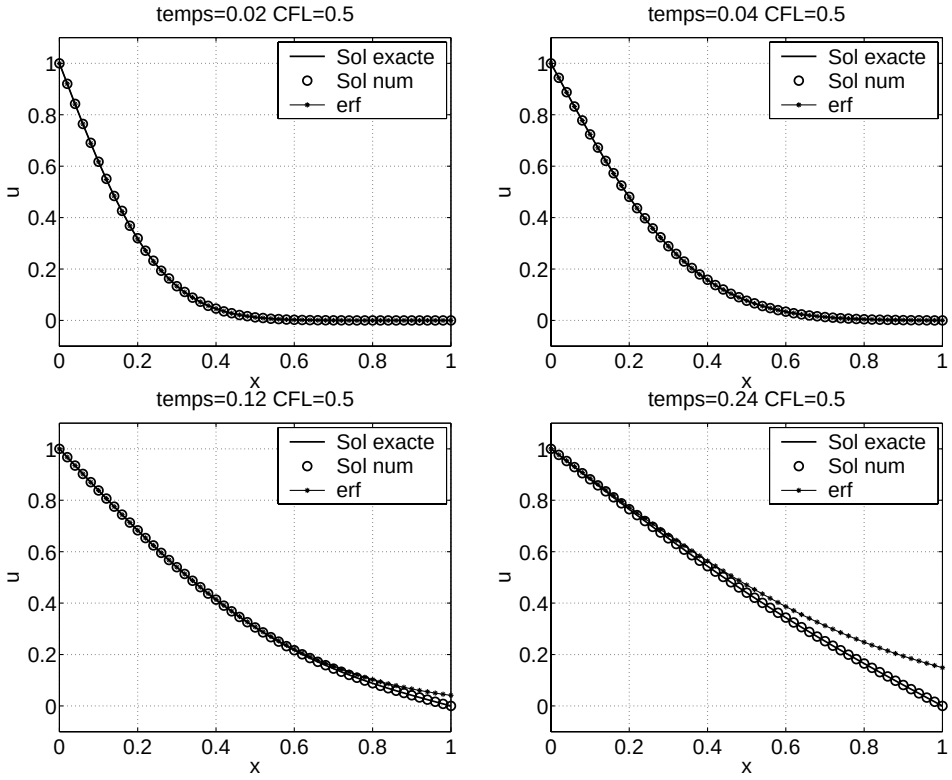


Figure 1.10 Solution numérique de l'équation de la chaleur pour $x \in [0, \ell]$, $\kappa = 1$ et les conditions aux limites $u(0,t) = 1, u(\ell,t) = 0$. Condition initiale $u(0,0) = 1, u(x,0) = 0, x > 0$. Comparaison avec la solution exacte (1.58) et la solution erf (1.57) obtenue pour un domaine infini.

Il est facile de revenir au programme précédent pour implémenter la condition initiale (1.61) - les lignes à changer sont marquées en commentaire. Les résultats (voir la figure 1.11) montrent clairement que les ondes avec un grand nombre d'onde (ou hautes fréquences) sont d'abord amorties ($k = 10$ dans notre cas). La solution tend vers la solution stationnaire constante $u(x) = 0$. Rappelons que l'équation des ondes transportait à vitesse constante la solution initiale sans modifier son amplitude (voir la figure 1.9).

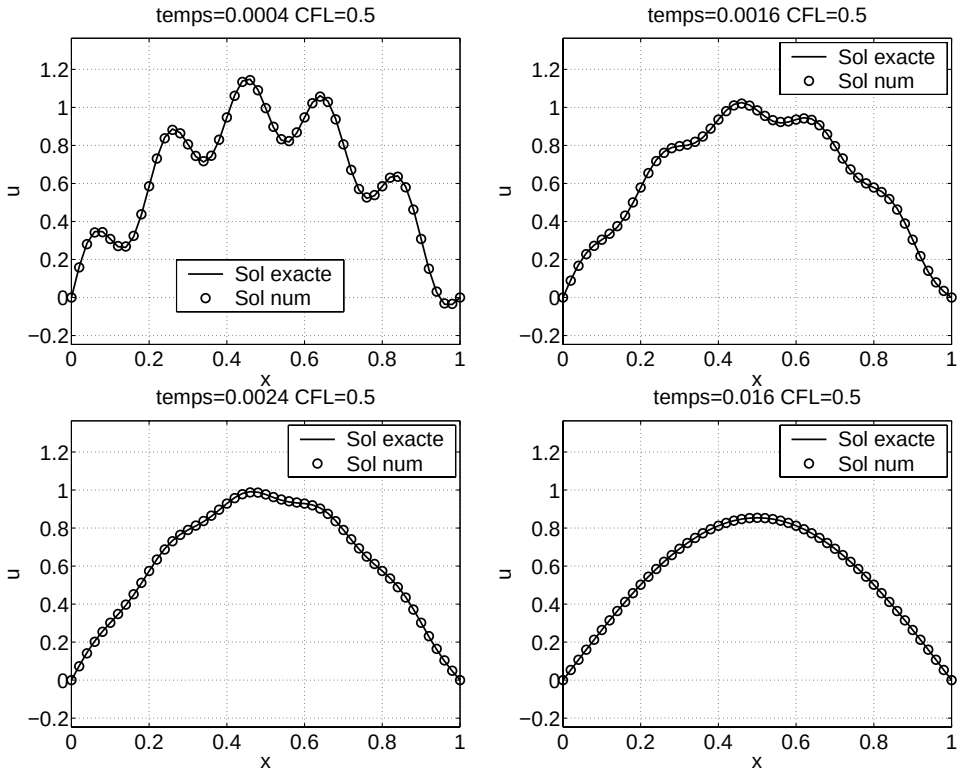


Figure 1.11 Solution numérique de l'équation de la chaleur pour $x \in [0, \ell]$, $\kappa = 1$ et les conditions aux limites $u(0,t) = 0, u(\ell,t) = 0$. Condition initiale : $u(x,0) = \sin(\pi x) + \sin(10\pi x)/4$. Comparaison avec la solution exacte (1.58). Observer l'amortissement des ondes de grande fréquence d'abord.

BIBLIOGRAPHIE

[Crouzeix et Mignot, 1989] M. Crouzeix, A.L. Mignot : *Analyse numérique des équations différentielles*, Masson, Paris, 1989.

[Danaila, Hecht et Pironneau, 2003] I. Danaila, F. Hecht, O. Pironneau : *Simulation numérique en C++*, Dunod, Paris, 2003.

[Delabrière et Postel, 2004] S. Delabrière, M. Postel *Méthodes d'approximation. Equations différentielles. Applications Scilab*, Ellipses, Paris, 2004.

[Demailly, 1996] J. P. Demailly : *Analyse numérique et équations différentielles*, Presses Universitaires de Grenoble, 1996.

[Hirsh, 1988] C. Hirsch : *Numerical computation of internal and external flows*, John Wiley & Sons, 1988.

[Lucquin et Pironneau, 1996] B. Lucquin, O. Pironneau : *Introduction au calcul scientifique*, Masson, 1996.

[Mohammadi et Saïac, 2003] B. Mohammadi, J.-H. Saïac : *Pratique de la simulation numérique*, Dunod, 2003.

Projet **2**

Équations différentielles non linéaires Application à la cinétique chimique

Fiche du projet

Difficulté :	1
Notions développées :	Système d'équations différentielles non-linéaires : stabilité, méthodes d'intégration, schéma d'Euler explicite, schéma de Runge-Kutta, équations différentielles à retard
Domaines d'application :	Cinétique chimique

2.1 PROBLÈME PHYSIQUE ET MODÉLISATION MATHÉMATIQUE

Les lois de la cinétique chimique peuvent s'écrire sous forme d'équations différentielles. Dans le cas de réactions complexes mettant en cause plusieurs molécules, les équations sont non linéaires et ont des propriétés intéressantes du point de vue mathématique (stabilité, périodicité, bifurcation, etc). La résolution numérique des équations différentielles est un domaine d'étude à part entière avec une littérature abondante. En ce qui concerne la mise en œuvre effective, dans MATLAB comme dans la plupart des logiciels de calcul scientifique, il existe des boîtes à outils efficaces dans la grande majorité des cas, dans la mesure où leur utilisateur maîtrise un

tant soit peut la théorie sous-jacente et les concepts de base tels que les notions de convergence, de stabilité, d'ordre (voir chapitre 1).

Le premier modèle de réaction que nous étudierons peut être résolu numériquement en utilisant un des solveurs d'équations différentielles standard. L'autre, comportant un terme de retard, nécessite l'implémentation d'un schéma numérique spécifique. Ils sont tous les deux inspirés d'exemples tirés de [Hairer, Norsett et Wanner, 1987].

On étudie en premier le modèle dit du $Brusselator$, mettant en jeu 6 substances A, B, D, E, X et Y

$$\left\{\begin{array}{rcll} A & \xrightarrow{v_1} & X & \\ B+X & \xrightarrow{v_2} & Y+D & \text{réaction bi-moléculaire} \\ 2X+Y & \xrightarrow{v_3} & 3X & \text{réaction tri-moléculaire auto-catalytique} \\ X & \xrightarrow{v_4} & E & \end{array}\right. \tag{2.1}$$

Où les v_i sont les vitesses (constantes) des différentes réactions chimiques. On dénote par $A(t)$, $B(t), \ldots$ les concentrations des substances $A, B, \ldots$. en fonction du temps t. On admettra que la conservation des masses dans les réactions chimiques ci-dessus se traduit par les équations différentielles

$$\left\{\begin{array}{l} A' = -v_1 A \\ B' = -v_2 BX \\ D' = v_2 BX \\ E' = v_4 X \\ X' = v_1 A - v_2 BX + v_3 X^2 Y - v_4 X \\ Y' = v_2 BX - v_3 X^2 Y \end{array}\right.$$

De ces six équations on peut éliminer dans un premier temps les deux équations donnant la production des espèces D et E, car ces espèces n'influencent pas l'évolution des quatre autres.

$$\left\{\begin{array}{l} A' = -v_1 A \\ B' = -v_2 BX \\ X' = v_1 A - v_2 BX + v_3 X^2 Y - v_4 X \\ Y' = v_2 BX - v_3 X^2 Y \end{array}\right.$$

Le système peut être simplifié en supposant que A et B sont conservés constants, et en prenant par ailleurs toutes les vitesses de réaction égales à 1. On obtient alors un système de deux équations à deux inconnues. Soit $U(t) = (X(t), Y(t))^T$, le vecteur modélisant les variations de concentration des substances X et Y, et $F(U) = (A - (B+1)X + X^2 Y, BX - X^2 Y)^T$, le système différentiel s'écrit sous la forme d'un problème de Cauchy

$$\left\{\begin{array}{l} U'(t) = F(U(t)), \\ U(0) = U_0 = (X_0, Y_0)^T. \end{array}\right. \tag{2.2}$$

2.2 ÉTUDE DE LA STABILITÉ DU SYSTÈME

On étudie la stabilité du système, c'est-à-dire sa propension à évoluer vers une solution $constante$ ou $stationnaire$. Cette solution stationnaire $U(t) = U_c$, si elle existe, vérifie $U'(t) = 0$, on la trouve donc en résolvant l'équation $F(U_c) = 0$ et on l'appelle $point\ critique$. Dans l'exemple ci-dessus la solution de cette équation est $U_c = (A, B/A)^T$. La stabilité du système caractérise son aptitude à revenir vers ce point critique en un temps fini, et on cherche donc une équation permettant d'étudier la variation $\Delta(t) = U(t) - U_c$. Pour cela, on linéarise F autour du point critique en écrivant la formule de Taylor

$$U'(t) = F(U) = F(U_c) + \nabla F_{U=U_c}(U - U_c) + \mathcal{O}(\|U - U_c\|^2)$$

avec

$$\nabla F = \begin{pmatrix} \dfrac{\partial F_1}{\partial X} & \dfrac{\partial F_1}{\partial Y} \\ \dfrac{\partial F_2}{\partial X} & \dfrac{\partial F_2}{\partial Y} \end{pmatrix} = \begin{pmatrix} 2XY - (B+1) & X^2 \\ B - 2XY & -X^2 \end{pmatrix}.$$

Pour étudier les petites variations de $\Delta(t)$, on néglige le terme en $\mathcal{O}(\|U - U_c\|^2)$ et on étudie le système différentiel linéaire

$$\begin{cases} \Delta'(t) = J\Delta(t), \\ \Delta(0) = \Delta_0, \end{cases} \tag{2.3}$$

où la matrice jacobienne $J = \nabla F_{U=U_c}$ vaut, dans le cas traité,

$$J = \begin{pmatrix} B - 1 & A^2 \\ -B & -A^2 \end{pmatrix}.$$

Dans le cas où J est diagonalisable, on l'écrit sous la forme $J = MDM^{-1}$. On a donc pour tout $n > 0$, $J^n = MD^nM^{-1}$. On rappelle la définition de l'exponentielle de la matrice J (voir par exemple [Allaire et Kaber, 2002 (1)])

$$e^J = \sum_{n=0}^{\infty} \frac{1}{n!} J^n = \sum_{n=0}^{\infty} \frac{1}{n!} MD^nM^{-1} = M \left(\sum_{n=0}^{\infty} \frac{1}{n!} D^n \right) M^{-1} = Me^DM^{-1},$$

où e^D est la matrice diagonale $(e^D)_{ij} = \delta_{ij} e^{\lambda_i}$, formée par les exponentielles des valeurs propres de J. Avec cette définition, le système différentiel (2.3) s'intègre directement :

$$\Delta(t) = e^{tJ}\Delta_0.$$

Dans ce cas le cas où la matrice J est diagonalisable, l'expression de la solution exacte calculée à la question précédente nous donne le comportement en temps long en faisant tendre t vers $+\infty$. Si toutes les valeurs propres λ de J sont de partie réelle négative, $e^{\lambda t} \to 0$ quand $t \to +\infty$, donc la matrice $e^{Jt} = Me^{Dt}M^{-1} \to 0$ quand $t \to +\infty$ et la solution ΔV tend vers 0. Le développement de Taylor autour du point critique est donc légitime, et la solution du système initial, non linéaire, tendra vers le point critique.

Dans ce cas très simple, on peut calculer explicitement les valeurs propres de la matrice J en cherchant les racines du polynôme caractéristique. On laisse au lecteur le soin de vérifier qu'elles sont égales à

$$\lambda_{\pm} = \frac{B - A^2 - 1 \pm \sqrt{\Delta}}{2}$$

et que leur partie réelle est négative si $B < A^2 + 1$.

On peut aussi utiliser une méthode numérique - qui nous donnera un critère de stabilité approché, mais utilisable même quand les valeurs propres sont difficiles à calculer explicitement. Pour cela, on propose l'exercice suivant

Exercice 2.1

> Écrire un programme représentant graphiquement la partie réelle maximale des valeurs propres de la matrice J en fonction du paramètre B pour une valeur constante du paramètre A. Représenter par un symbole la valeur exacte du critère de stabilité.

La solution de cet exercice se trouve en page 39.

Pour résoudre le système d'équations différentielles (2.2), on pourrait utiliser un des schémas d'intégration proposé dans le chapitre 1 (voir Table 1.1). Une alternative est de laisser le logiciel faire le travail et utiliser un de ses solveurs d'équations différentielles.

Exercice 2.2

> Calculer numériquement les solutions approchées pour des valeurs de A et B correspondant aux cas de stabilité et d'instabilité. Dans chaque cas, représenter les solutions X et Y en fonction du temps, et sur un autre graphique Y en fonction de X.

La solution de cet exercice se trouve en page 40.

2.3 MODÈLE DE LA RÉACTION ENTRETENUE

On reconsidère maintenant le système initial en faisant cette fois l'hypothèse que le composant B n'est plus constant mais injecté dans le mélange à la vitesse v. On note $Z(t)$ la concentration en B en fonction du temps. On admettra que le système des réactions chimiques (2.1) se ramène alors, pour $A = 1$, à un nouveau système de trois équations différentielles

$$\begin{cases} X' = 1 - (Z+1)X + X^2 Y \\ Y' = XZ - X^2 Y \\ Z' = -XZ + v \end{cases} \tag{2.4}$$

2.3.1 Existence d'un point critique et stabilité

Le problème (2.4) admet maintenant une solution stationnaire au point critique $U_c = (1, v, v)^T$.

La matrice jacobienne de la fonction second membre du système (2.4) est

$$\nabla F = \begin{pmatrix} -(Z+1) + 2XY & Z - 2XY & -Z \\ X^2 & -X^2 & 0 \\ -X & X & -X \end{pmatrix}.$$

Pour étudier la stabilité du système, on évalue cette matrice au point critique :

$$J = \begin{pmatrix} v-1 & 1 & -1 \\ -v & -1 & 1 \\ -v & 0 & -1 \end{pmatrix}.$$

Exercice 2.3

> Trouver numériquement des valeurs de v pour lesquelles le système est stable ou instable. On s'inspirera de la méthode numérique proposée à la question (2.1).

La solution de cet exercice se trouve en page 41.

2.3.2 Résolution numérique

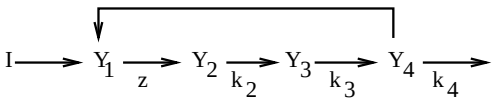
Exercice 2.4

> Résoudre numériquement le système (2.4) pour les valeurs de v suivantes : 0.9, 1.3 et 1.52.
>
> Dans chaque cas, tracer sur des figures différentes les trois concentrations en fonction du temps puis deux concentrations en fonction de la troisième.

La solution de cet exercice se trouve en page 41.

2.4 MODÈLE DE LA RÉACTION AVEC RETARD

Un exemple de réaction chimique plus compliquée est proposé dans [Hairer, Norsett et Wanner, 1987]. Une quantité de substance I est introduite à flux constant dans le système, produisant une réaction en chaîne



La quantité de produit final Y_4 intervient dans l'étape $Y_1 \to Y_2$ en la ralentissant, et une modélisation fine de ce processus, prenant en compte le temps de transport et de diffusion des molécules, conduit à un système d'équations différentielles *à retard* :

$$\left\{\begin{array}{rcl} y_1'(t) &=& I - z(t)y_1(t) \\ y_2'(t) &=& z(t)y_1(t) - y_2(t) \\ y_3'(t) &=& y_2(t) - y_3(t) \\ y_4'(t) &=& y_3(t) - 0.5y_4(t) \\ z(t) &=& \dfrac{1}{1+\alpha y_4(t-4)^3}. \end{array}\right. \tag{2.5}$$

Ce système a un point critique Y_c, obtenu en résolvant $y'(t) = 0$:

$$Y_c = \begin{pmatrix} I(1+8\alpha I^3) \\ I \\ I \\ 2I \end{pmatrix} \tag{2.6}$$

Comme au paragraphe précédent, on peut linéariser le système autour de ce point, et étudier la stabilité du système linéaire : on introduit une cinquième variable $y_5(t) = y_4(t-4)$ et on calcule le Jacobien de la fonction

$$F(y) = \begin{pmatrix} I - \dfrac{y_1}{1+\alpha y_5^3} \\ \dfrac{y_1}{1+\alpha y_5^3} - y_2 \\ y_2 - y_3 \\ y_3 - 0.5y_4 \end{pmatrix}.$$

On obtient la matrice jacobienne

$$\nabla F = \begin{pmatrix} -\overline{z} & 0 & 0 & 0 & 12\alpha I^3\overline{z} \\ \overline{z} & -1 & 0 & 0 & -12\alpha I^3\overline{z} \\ 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 \end{pmatrix},$$

avec $\overline{z} = \dfrac{1}{1+8\alpha I^3}$. Les petites variations $\Delta(t)$ autour du point critique Y_c vérifient en première approximation le système linéaire :

$$\begin{array}{rcl} \Delta_1'(t) &=& -\overline{z}\Delta_1(t) + 12\alpha I^3\overline{z}\Delta_4(t-4), \\ \Delta_2'(t) &=& \overline{z}\Delta_1(t) - \Delta_2(t) - 12\alpha I^3\overline{z}\Delta_4(t-4), \\ \Delta_3'(t) &=& \Delta_2(t) - \Delta_3(t), \\ \Delta_4'(t) &=& \Delta_3(t) - 0.5\Delta_4(t). \end{array}$$

On cherche $\Delta(t)$ sous la forme $\Delta(t) = Ve^{xt}$, avec $V \in \mathbb{R}^4$ un vecteur constant. On obtient l'équation caractéristique permettant de calculer x

$$(x+1)^2(x+0.5)(x+\overline{z}) + 12\alpha\overline{z}I^3xe^{-4x} = 0.$$

Le système correspondant sera stable si toutes les racines de l'équation ont une partie réelle négative.

Exercice 2.5

> On fixe $\alpha = 0.0005$. Résoudre numériquement cette équation dans $\mathbb{C}$ pour différentes valeurs de I entre 0 et 20. Trouver numériquement une valeur minimale pour le paramètre I au delà de laquelle on peut exhiber des solutions correspondant à un équilibre instable.

La solution de cet exercice se trouve en page 43.

Pour illustrer numériquement le phénomène d'instabilité, il faut intégrer le système complet (2.5), et pour ce faire, on ne peut pas utiliser un solveur standard prévu pour les équations différentielles ordinaires du type

$$\begin{cases} y'(t) = F(t, y(t)), \\ y(0) = u_0, \end{cases} \tag{2.7}$$

alors qu'ici nous avons une dépendance par rapport aux valeurs antérieures de la solution

$$\begin{cases} y'(t) = G(t, y(t), y(t-4)), \\ y(t) = u_0(t), \quad \text{pour} \quad t \leqslant 0. \end{cases} \tag{2.8}$$

Dans notre exemple, la fonction G est la fonction vectorielle de $\mathbb{R} \times \mathbb{R}^4 \times \mathbb{R}^4$ dans $\mathbb{R}^4$

$$G(t,u,v) = \begin{pmatrix} I - \dfrac{u_1}{1+\alpha v_4^3} \\ \dfrac{u_1}{1+\alpha v_4^3} - u_2 \\ u_2 - u_3 \\ u_3 - \dfrac{1}{2}u_4 \end{pmatrix}. \tag{2.9}$$

Il s'agit donc d'adapter les schémas numériques prévus pour résoudre les systèmes du type (2.7). Prenons par exemple le cas le plus simple du schéma d'Euler explicite

$$\begin{array}{l} \text{initialisation :} \quad y_0 = u_0 \\ \text{pour} \quad i = 0, 2, \ldots, n-1 \quad \text{faire} \\ \qquad y_{i+1} = y_i + hF(t_i, y_i) \\ \text{fin de} \quad i \end{array} \tag{2.10}$$

Ce schéma permet de calculer l'approximation $y_n \approx y(t_n)$, avec $t_n = nh$ pour h suffisamment petit (voir [Delabrière et Postel, 2004]).

Si le délai (ici 4) est un multiple entier du pas de temps h, $4 = dh$ avec d entier, le schéma s'adapte facilement au système avec retard (2.8)

$$\begin{array}{l} \text{initialisation :} \quad y_0 = u_0(0) \\ \text{pour} \quad i = 0, 1, 2, \ldots, n-1 \quad \text{faire} \\ \qquad \gamma_i = \begin{cases} u_0(t_i - 4) & \text{si} \quad i < d \\ y_{i-d} & \text{sinon} \end{cases} \\ \qquad y_{i+1} = y_i + hG(t_i, y_i, \gamma_i) \\ \text{fin de} \quad i \end{array} \tag{2.11}$$

Exercice 2.6

1. On suppose que la solution est constante égale à y_0 pour tout $t \leqslant 0$. Écrire une fonction `RetardEnzyme(t,Y,h,y0)` renvoyant la valeur $G(t, y(t), y(t-4))$ (2.9) quand les valeurs de y sont discrétisées dans le tableau Y avec le pas $h = t_{max}/n_{max}$.
2. Écrire une fonction `EulerRetard(fretard,tmax,n,y0)` mettant en œuvre l'algorithme (2.11) pour calculer une approximation de la solution au temps `tmax` en `n` pas de temps.
3. Écrire un programme principal pour intégrer le système avec l'algorithme (2.11) jusqu'à $t_{max} = 160$. On fixe $\alpha = 0.0005$ et on choisit pour I une valeur correspondant à un cas d'instabilité. Choisir une condition initiale y_0 proche de la solution d'équilibre instable y_C. Représenter graphiquement les 4 composantes de la solution sur la même figure, et dans une autre fenêtre les composantes y_i, $i = 2, \ldots, 4$ en fonction de la composante y_1.

La solution de cet exercice se trouve en page 44.

Dans le cas d'un schéma de Runge-Kutta , il faut conserver les valeurs intermédiaires intervenant dans le calcul de y_{i+1} en fonction de y_i. Pour adapter le schéma d'ordre 4 présenté au chapitre 1, on l'écrit maintenant de manière à faire apparaître les arguments intermédiaires de la fonction second membre, plutôt que la valeur du second membre en ces points.

$$\begin{array}{l} \text{initialisation :} \quad y_0 = u_0 \\ \text{pour} \quad i = 0, 2, \ldots, n-1 \quad \text{faire} \\ \qquad g^1 = y_i, \\ \qquad g^2 = y_i + \dfrac{h}{2} F(t_i, g^1), \\ \qquad g^3 = y_i + \dfrac{h}{2} F(t_i, g^2), \\ \qquad g^4 = y_i + hF(t_i, g^3), \\ \qquad y_{i+1} = y_i + \dfrac{h}{6}\left(F(t_i, g^1) + 2F(t_i + \dfrac{h}{2}, g^2) + 2F(t_i + \dfrac{h}{2}, g^3) + F(t_i + h, g^4) \right). \\ \text{fin de} \quad i \end{array} \tag{2.12}$$

Il faut conserver les valeurs g^k, $k = 1, \ldots, 4$ en fonction du temps, pour pouvoir calculer les valeurs intermédiaires avec le délai d :

initialisation : $y_0 = u_0$
pour $i = 0, 2, \ldots, n-1$ faire
$$g_i^1 = y_i,$$
$$g_i^2 = y_i + \frac{h}{2} G(t_i, g_i^1, \gamma_i^1),$$
$$g_i^3 = y_i + \frac{h}{2} G(t_i, g_i^2, \gamma_i^2),$$
$$g_i^4 = y_i + hG(t_i, g_i^3, \gamma_i^3),$$
$$y_{i+1} = y_i + \frac{h}{6}\left(G(t_i, g_i^1, \gamma_i^1) + 2G(t_i + \frac{h}{2}, g_i^2, \gamma_i^2) + 2G(t_i + \frac{h}{2}, g_i^3, \gamma_i^3) + G(t_i + h, g_i^4, \gamma_i^4) \right),$$
où $\gamma_i^k = \begin{cases} u_0(t_i + c_k h - 4) & \text{si } i < d, \\ g_{i-d}^k & \text{sinon,} \end{cases}$
avec $c = \begin{pmatrix} 0 & 0.5 & 0.5 & 1 \end{pmatrix}^T$.
fin de i

(2.13)

Exercice 2.7

1. Écrire une fonction `RungeKuttaRetard(fretard,tmax,n,y0)` mettant en œuvre l'algorithme (2.13) pour calculer une approximation de la solution au temps `tmax` en `n` pas de temps.
2. Comparer graphiquement les solutions obtenues avec les algorithmes d'Euler et de Runge-Kutta d'ordre 4. On fixe maintenant $t_{max} = 16$. Représenter les deux solutions calculées pour $n_{max} = 100$ puis pour $n_{max} = 1000$. En utilisant la solution calculée avec $n_{max} = 5000$ comme solution "exacte", calculer l'erreur en norme L^∞ en fonction de h, pour n_{max} variant entre 100 et 2000.
3. Étudier l'influence de la condition initiale : tracer les trajectoires y_i, $i = 2, \ldots, 4$ en fonction de y_1 avec des couleurs différentes pour des conditions initiales différentes.

La solution de cet exercice se trouve en page 44.

2.5 SOLUTIONS ET PROGRAMMES

Tous les programmes et fonctions MATLAB proposés en solution des exercices sont disponibles sur le site web du livre.

Solution de l'exercice 2.1

Pour illustrer graphiquement le critère de stabilité, on propose le script `stab2comp.m` : on calcule numériquement les valeurs propres du Jacobien de F au point critique,

à l'aide de la fonction MATLAB `eigen` et on calcule la valeur maximale de la partie réelle de ces valeurs propres. Pour A fixé, on trace cette valeur en fonction du paramètre B, et on peut lire la valeur approximative du critère de stabilité qui est l'abscisse où cette valeur maximale change de signe.

Solution de l'exercice 2.2

Pour intégrer numériquement le système (2.2) on propose le script `Chimie2.m` qui utilise le solveur d'équations différentielles `ode45` développé par MATLAB.

```
global A
global B
fun='fun2';
A=1;
B=0.9;
%
U0=[2;1]; % Condition  initiale
t0=0;     % temps initial
t1=10;   % temps final
[tempsS1,solS1]=ode45(fun,[t0,t1],U0);
```

L'extrait ci-dessus correspond à un cas stable. La fonction MATLAB `ode45` prend en argument d'entrée la fonction second membre du système différentiel programmée dans `fun2.m`, l'intervalle de temps `[t0,t1]` sur lequel on intègre le système, et la solution `U0` à l'instant initial `t0`. Elle renvoie en sortie le tableau `tempsS1` des temps où le solveur à calculé la solution dans le tableau `solS1`. Les paramètres `A` et `B` du système différentiel sont déclarés `global` dans le script d'appel et dans la fonction second membre `fun2`, ce qui évite d'avoir à les passer en arguments d'appel de cette dernière.

Pour $A = 1$, on fait tourner le programme avec des jeux de paramètres correspondant à la stabilité $B = 0.9$ ou l'instabilité $B = 3.2$. Dans le premier cas, les concentrations tendent vers une valeur constante qui est justement celle du point critique. On représente ce comportement sur la figure 2.1. On voit à gauche l'allure des concentrations en fonction du temps. Elles se stabilisent rapidement sur leurs valeurs critiques. Le graphe de droite montre le comportement du constituant Y en fonction du constituant X pour deux conditions initiales différentes. Les deux trajectoires convergent toutes les deux vers le point critique de coordonnées $(A, B/A) = (1, 0.9)$. Dans le deuxième cas, elles restent bornées mais ont une allure périodique en fonction du temps . Si on trace Y en fonction de X on voit, en regardant le phénomène en temps long, le cycle limite. Le graphe de droite de la figure 2.2 illustre numériquement que ce cycle ne dépend pas des concentrations initiales mais juste des valeurs des paramètres. En se rapprochant de la limite d'instabilité, $B = A^2 + 1$ on observe que ce cycle limite devient de plus en plus petit, pour disparaître finalement au point critique. Ce phénomène s'appelle *bifurcation de Hopf*.

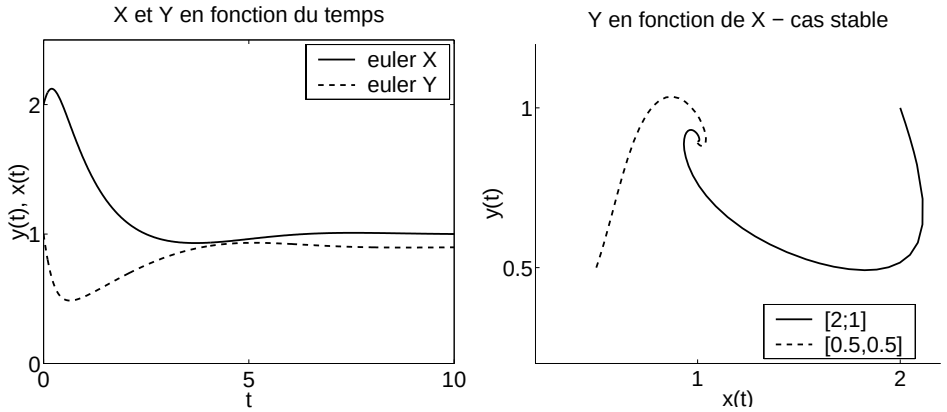


Figure 2.1 Modèle du Brusselator simplifié, cas stable $A = 1$, $B = 0.9$. À gauche, les concentrations X et Y en fonction du temps. À droite, les courbes paramétrées $(X, Y)_t$ pour deux conditions initiales différentes $(2, 1)^T$ et $(0.5, 0.5)^T$.

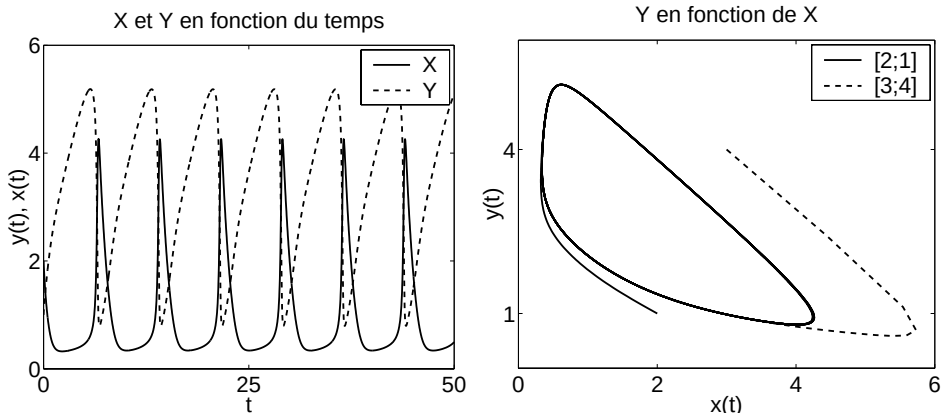


Figure 2.2 Modèle du Brusselator simplifié, cas instable $A = 1$, $B = 3.2$. À gauche, les concentrations X et Y en fonction du temps. À droite, les courbes paramétrées $(X, Y)_t$ pour deux conditions initiales différentes $(2, 1)^T$ et $(2, 3)^T$.

Solution de l'exercice 2.3

On adapte le script `stab2comp.m` mis au point dans la question 2.1 pour trouver les valeurs du paramètre v pour lesquelles toutes les valeurs propres de J ont des parties réelles négatives. On voit sur la figure produite par le script `stab3comp.m`, que pour les valeurs 0.9, 1.3 et 1.52 proposées dans l'exercice 2.4 il y aura stabilité seulement pour la première.

Solution de l'exercice 2.4

L'intégration du système de trois équations est faite dans le script `Chimie3.m`. La fonction second membre du système est définie dans `fun3.m` Pour $v = 0.9$, le système est stable et donc les concentrations tendent toutes les trois vers leur valeur d'équilibre ($U_c = (1, 0.9, 0.9)^T$) comme le montre le graphe de gauche de la figure (2.3). Le graphe de droite, qui montre les variations de Y en fonction de X,

permet aussi de visualiser la convergence vers le point critique, à partir de plusieurs conditions initiales différentes.

Pour $v = 1.3$, le système est instable, mais les concentrations restent bornées. Leur variation en fonction du temps est périodique et tend vers un cycle limite comme sur la figure (2.4)

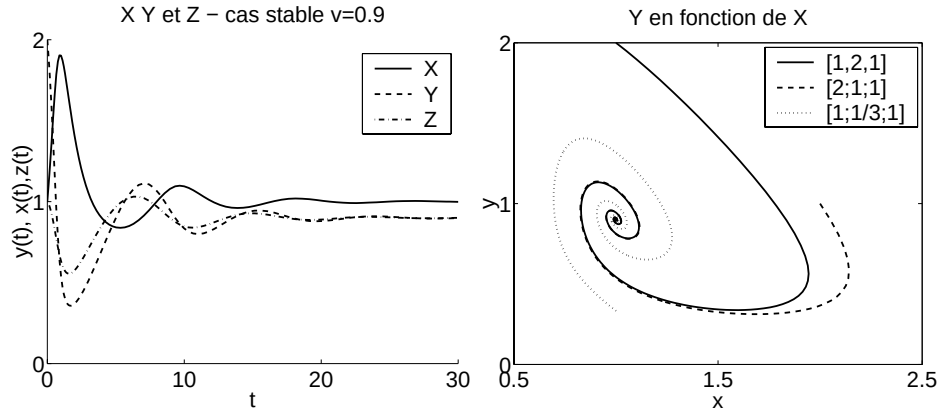


Figure 2.3 Modèle du Brusselator, cas stable $v = 0.9$. À gauche, les concentrations X, Y et Z en fonction du temps. À droite, les courbes paramétrées $(X, Y)_t$ pour deux conditions initiales différentes $(1, 2, 1)^T$ et $(2, 2, 2)^T$.

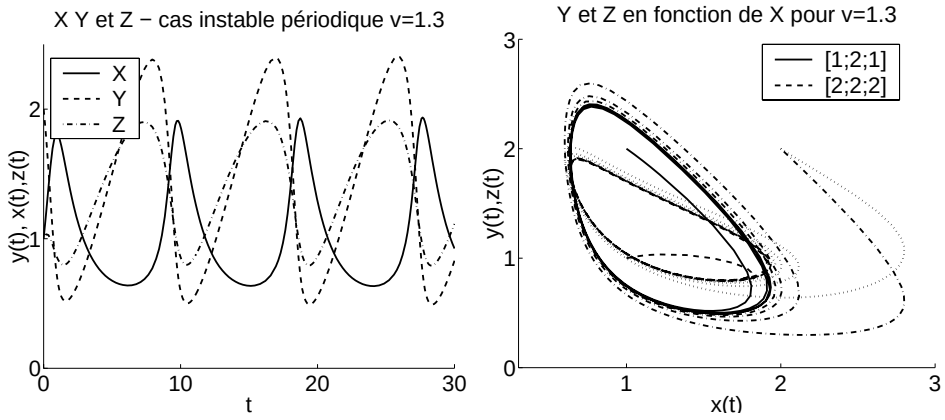


Figure 2.4 Modèle du Brusselator, cas instable périodique $v = 1.3$. À gauche, les concentrations X, Y et Z en fonction du temps. À droite, la courbe paramétrée $(X, Y)_t$ pour deux conditions initiales différentes $(1, 2, 1)^T$ et $(2, 2, 2)^T$.

Enfin, pour $v > 1.5$ le système est instable ou divergent et les concentrations y et z explosent alors que la concentration x tend vers 0. Le comportement est alors complètement différent du précédent. Il n'y a pas de cycle limite de y en fonction de x, ou de z en fonction de y.

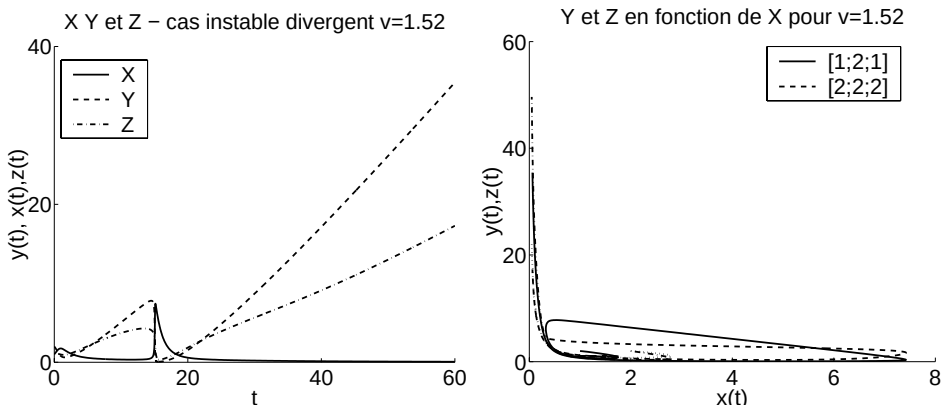


Figure 2.5 Modèle du Brusselator, cas instable divergent $v = 1.52$. À gauche, les concentrations X, Y et Z en fonction du temps. À droite, les courbes paramétrées $(X, Y)_t$ pour deux conditions initiales différentes $(1, 2, 1)^T$ et $(2, 2, 2)^T$.

Solution de l'exercice 2.5

Cette équation peut se résoudre numériquement avec MATLAB en utilisant la fonction `fsolve`, comme dans le script suivant

Listing 2.1 *StabRetard.m*

```
for  I=0:20
 alpha=0.0005;
 barz=1/(1+8*alpha*I^3);
 funequi=inline('(x+1)^2*(x+0.5)*(x+barz)+12*alpha*I^3*barz*x*exp(-
4*x)',...
         'x','barz','I','alpha');
 guess=i/2;
 x0=fsolve(funequi,guess,optimset('Display','off'),barz,I,alpha);
 fprintf('I=%f x0=%f+i%f\n',I,real(x0),imag(x0))
end
```

Si on fait tourner le script avec une valeur initiale réelle `guess=2` par exemple, la solution trouvée par le solveur est toujours réelle et négative donc correspond à un équilibre stable puisque les déviations à partir du point critique tendent exponentiellement vers 0. En revanche, il suffit de faire tourner ce script avec une valeur initiale imaginaire pure pour obtenir une racine imaginaire. En choisissant comme ci-dessus `guess=i/2` on obtient une solution avec une partie réelle positive à partir de $I = 9$. la solution d'équilibre correspondant à un tel choix de paramètres est donc instable puisqu'alors les déviations augmentent exponentiellement. En revanche, on n'est pas sûr d'avoir la stabilité pour $I < 9$ puisque nous n'avons pas montré que toutes les solutions ont leur partie réelle négative.

Solution de l'exercice 2.6

Le script principal `Enzyme.m` fait appel aux fonctions `RetardEnzyme` et `EulerRetard`. On choisit `I=10` ce qui correspond à un cas d'instabilité La condition initiale est fixée en ajoutant une petite déviation à la solution d'équilibre instable (2.6). On remarque sur la figure 2.6 que les trajectoires se superposent, ce qui indique que la solution devient périodique. On peut d'ailleurs évaluer graphiquement la période (environ 13), qui correspond bien à une des phases déterminées en résolvant l'équation caractéristique.

En fixant `I=5` dans le script, valeur pour laquelle la recherche d'une solution linéaire instable a échoué, on remarque au contraire que les solutions tendent vers le point d'équilibre.

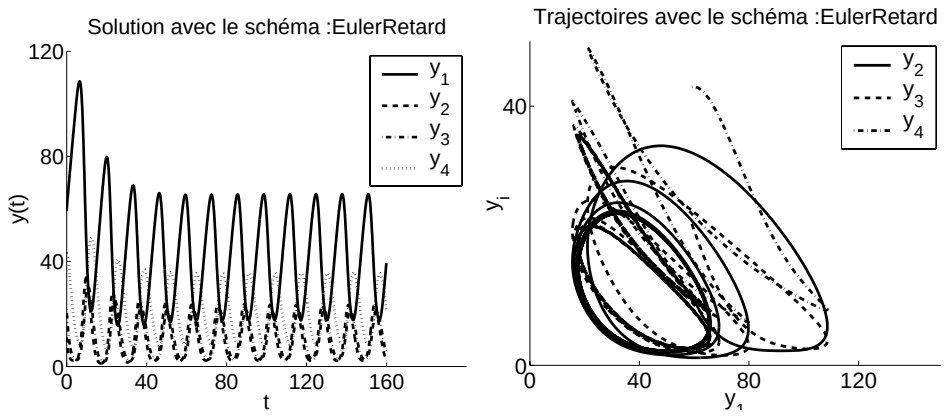


Figure 2.6 Solutions du système (2.5) obtenues avec le schéma d'Euler. $y(t)$ à gauche et trajectoires y_i, $i = 1, \ldots, 3$ en fonction de y_1 à droite.

Solution de l'exercice 2.7

Pour refaire la simulation en utilisant cette fois-ci le schéma de Runge-Kutta d'ordre 4, on remplace `EulerRetard` par `RungeKuttaRetard` dans le script `Enzyme.m`. Pour implémenter le schéma de Runge-Kutta adapté aux équations différentielles à retard (2.13), on introduit 4 tableaux `G( :,n,k)` pour $k = 1, \ldots, 4$ pour stocker au cours du temps les 4 arguments intermédiaires de la fonction second membre (voir `RungeKuttaRetard.m`).

Une étude plus fine de l'ordre de convergence des deux schémas est proposée dans le script `ErreurEnzyme`. Les solutions de référence pour chaque schéma, calculées avec une discrétisation "très" fine, ici `nmax=5000`, sont utilisées comme solutions "exactes", pour évaluer l'erreur commise sur la solution au temps final, ici 50, avec des discrétisations plus grossières. Sur la figure 2.7, les variations de

l'erreur en fonction de h sont représentées en coordonnées logarithmiques avec les ordres théoriques $O(h)$ et $O(h^4)$ ce qui permet leur comparaison.

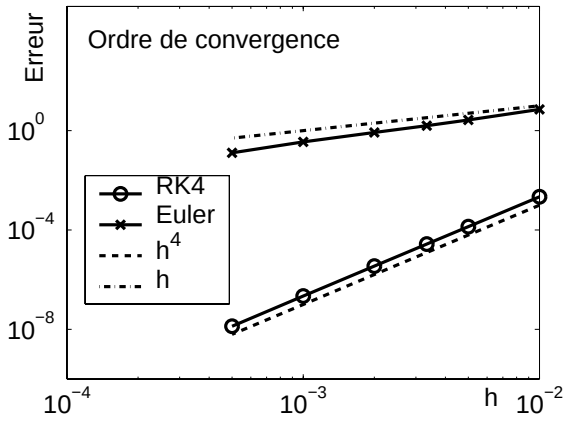


Figure 2.7 Erreur en norme L^∞ au temps $t = 50$ pour les schémas d'Euler et de Runge-Kutta d'ordre 4.

Enfin, l'influence de la condition initiale est illustrée par le script `EnzymeCondIni` qui représente sur le même graphe les trajectoires issues de conditions initiales différentes, choisies aléatoirement. On voit sur la figure 2.8 qu'après une phase initiale plus ou moins courte, elles convergent toutes vers la même trajectoire périodique.

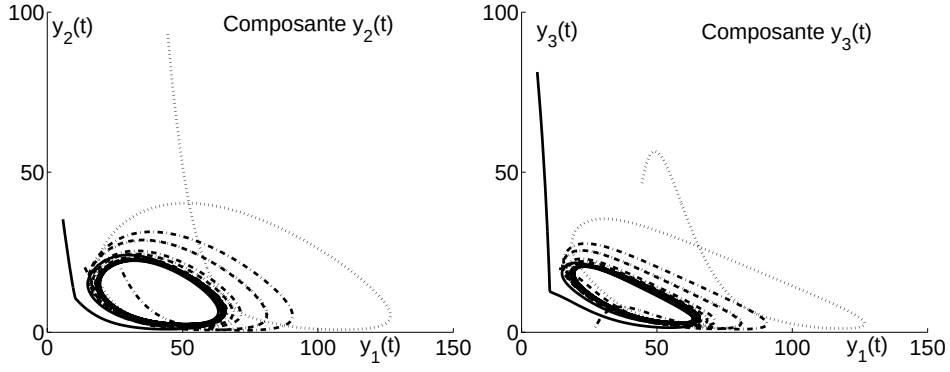


Figure 2.8 Solutions du système (2.5) obtenues pour 4 conditions initiales différentes, avec le schéma de Runge-Kutta. Trajectoires y_2 en fonction de y_1 à gauche et y_3 en fonction de y_1 à droite.

BIBLIOGRAPHIE

[Allaire et Kaber, 2002 (1)] G. ALLAIRE, S. M. KABER *Cours d'Algèbre Linéaire Théorique.* Ellipses, Paris (2002).

[Delabrière et Postel, 2004] S. DELABRIÈRE, M. POSTEL *Méthodes d'Approximation. Équations Différentielles. Applications Scilab.* Ellipses, Paris (2004).

[Hairer, Norsett et Wanner, 1987] E. HAIRER, S.P. NORSETT, G. WANNER *Solving Ordinary Differential Equations I, Nonstiff Problems.* Springer series in computational mathematics, 8, Springer Verlag (1987).

Projet **3**

Approximation polynomiale

Fiche du projet

Difficulté :	1
Notions développées :	Approximation polynomiale, splines, calcul du polynôme d'interpolation, calcul de meilleures approximations polynomiales
Domaines d'application :	Représentation de fonctions

Ce chapitre est consacré au problème de l'approximation d'une fonction numérique par une fonction plus « simple », appartenant par exemple à $\mathbb{P}_n$, ensemble des polynômes de degré inférieur ou égal à n. On considérera aussi l'approximation par des fonctions polynomiales par morceaux, c'est-à-dire dont la restriction à des intervalles est un polynôme. Les notions et résultats de ce chapitre, énoncés sans démonstration, sont utilisés dans l'ensemble du livre. Le lecteur est invité à lire des ouvrages traitant de l'approximation polynomiale pour les démonstrations, en particulier [Crouzeix et Mignot, 1989] et [Dumas, 1999].

3.1 INTRODUCTION

L'approximation d'une fonction par un polynôme peut être utilisée pour résoudre les problèmes suivants.

1. *Visualisation de résultats de calculs.* On connaît les valeurs de la fonction f en des points x_i et on cherche à représenter f sur un intervalle $[a, b]$. C'est le problème de l'interpolation (si $[a, b] \subset [\min_i x_i, \max_i x_i]$) ou de l'extrapolation (si l'un des points a ou b n'appartient pas à $[\min_i x_i, \max_i x_i]$). On fait alors l'approximation
$$\forall x \in [a, b] \qquad f(x) \simeq p_n(x).$$
2. *Intégration numérique.* Pour calculer l'intégrale de la fonction f sur un intervalle $[a, b]$, on fait l'approximation
$$\int_a^b f(x)dx \simeq \int_a^b p_n(x)dx.$$
La dernière intégrale pouvant être calculée de façon exacte.
3. *Équations différentielles.* Pour résoudre numériquement ces équations, on peut remplacer les dérivées de f par les dérivées de p_n, puis résoudre l'équation différentielle vérifiée par p_n. Voir le chapitre 5.

3.2 APPROXIMATION POLYNOMIALE

Approcher f par $p_n \in \mathbb{P}_n$ peut signifier :

Interpolation. Le polynôme p_n et la fonction f coïncident en $n+1$ points $x_0, \ldots, x_n$ de l'intervalle $[a, b]$. Ces points peuvent être imposés ou être eux-mêmes des inconnues du problème.

Meilleure approximation. Le polynôme p_n est l'élément (ou un élément) de $\mathbb{P}_n$ (s'il existe) le plus proche de f pour une norme $\|.\|$ donnée. Plus précisément
$$\|f - p_n\| = \inf_{q \in \mathbb{P}_n} \|f - q\|.$$
Si par exemple, la norme choisie est $\|\varphi\|_2 = \sqrt{\int_a^b |\varphi(x)|^2 dx}$, on parle d'approximation au sens des moindres carrés ou au sens L^2 ou encore meilleure approximation hilbertienne. Si la norme en question est la norme de la convergence uniforme, qui sera notée dans la suite $\|\varphi\|_\infty = \sup_{x \in [a,b]} |\varphi(x)|$, on parle d'approximation au sens de la convergence uniforme ou encore au sens L^∞.

3.2.1 Interpolation polynomiale

Dans cette section $f : [a, b] \longrightarrow \mathbb{R}$ est une fonction continue, $(x_i)_{i=0}^k$ un ensemble de $k+1$ points *distincts* donnés de l'intervalle réel $[a, b]$ et $(\alpha_i)_{i=0}^k$ un ensemble de $(k+1)$

entiers. On pose $n = k + \alpha_0 + \ldots + \alpha_k$. Nous nous intéressons au problème suivant : existe-t-il un polynôme p qui coïncide avec f ainsi qu'éventuellement certaines de ses dérivées aux points x_i ?

a) Interpolation de Lagrange

Elle correspond au cas où on cherche à interpoler une fonction, mais pas ses dérivées. On a alors $\alpha_i = 0$ pour tout i et donc $k = n$. La théorie nous apprend que

Théorème 3.1 *Pour tout choix de* $(n + 1)$ *points distincts* $x_0, x_1, \ldots, x_n$ *et pour toute fonction continue* f*, il existe un unique polynôme* $p_n \in \mathbb{P}_n$ *qui vérifie pour tout* $i = 0, \ldots, n$

$$p_n(x_i) = f(x_i). \tag{3.1}$$

Le polynôme p est appelé polynôme d'interpolation de Lagrange ou polynôme d'interpolation de f aux points x_i, on le notera $\mathcal{I}_n(f; x_0, \ldots, x_n)$ ou plus simplement $\mathcal{I}_n f$, s'il n'y a pas d'ambiguïté.

On appelle polynômes caractéristiques de Lagrange associés aux points x_i, les $n+1$ polynômes $(\ell_i)_{i=0}^n$ définis par

$$\ell_i \in \mathbb{P}_n \text{ et } \ell_i(x_j) = \delta_{i,j} \quad \text{pour } j = 0, \ldots, n.$$

L'intérêt de la base de Lagrange réside dans le fait que le polynôme d'interpolation de Lagrange s'écrit simplement dans cette base

$$\mathcal{I}_n f = \sum_{i=0}^{n} f(x_i)\ell_i. \tag{3.2}$$

Les polynômes de Lagrange forment une base de $\mathbb{P}_n$ et sont explicitement donnés par

$$\ell_i(x) = \prod_{j=0, j\neq i}^{n} \frac{x - x_j}{x_i - x_j}. \tag{3.3}$$

À titre d'illustration, les quatre polynômes de Lagrange associés aux quatre points -1, 0, 1 et 3 sont représentés sur la figure 3.1.

➤ Calcul du polynôme d'interpolation de Lagrange.

La question est de déterminer la base de $\mathbb{P}_n$ la plus adaptée pour le calcul de $\mathcal{I}_n f$. Nous allons comparer trois bases

Base 1. La base canonique formée des monômes $1, x, \ldots, x^n$.

Base 2. La base formée des polynômes de Lagrange.

Base 3. La base formée des polynômes

$$1, (x - x_0), (x - x_0)(x - x_1), \ldots, (x - x_0)(x - x_1)\ldots(x - x_{n-1}). \tag{3.4}$$

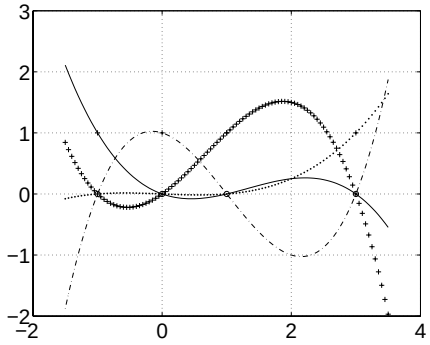


Figure 3.1 Polynômes de Lagrange associés aux points −1, 0, 1 et 3.

Exercice 3.1 Calculs dans la base canonique

Soient $(a_k)_{k=0}^n$ les coefficients de $\mathcal{I}_n f$ dans la base canonique et :

$$a = (a_0, \ldots, a_n)^T \in \mathbb{R}^{n+1} : \mathcal{I}_n f = \sum_{k=0}^{n} a_k x^k.$$

1. Montrer que les conditions d'interpolation (3.1) sont équivalentes à la résolution d'un système linéaire $Aa = b$, dont la matrice $A \in \mathbb{R}^{(n+1)\times(n+1)}$ et le second membre $b \in \mathbb{R}^{n+1}$ sont à déterminer.
2. Pour $n = 10$ (puis 20) créer un tableau x de $n + 1$ valeurs aléatoires triées par ordre croissant entre 0 et 1.
3. Écrire un programme calculant la matrice A.
4. On pose $f(x) = \sin(10\,x\cos x)$. Calculer les coefficients de $\mathcal{I}_n f$ en résolvant le système linéaire $A\alpha = b$. Représenter sur un même graphique $\mathcal{I}_n f$ et la fonction f aux points x_i. On utilisera la fonction `polyval`. (Attention à l'ordre des coefficients α_i).
5. Pour $n = 10$, calculer $\|A\alpha - b\|_2$, puis le conditionnement en norme 2 de la matrice A (fonction `cond`) et son rang (fonction `rank`). Mêmes questions pour $n = 20$. Commenter.

La solution de cet exercice se trouve page 70.

Exercice 3.2

Calculer le logarithme du conditionnement de la matrice A de l'exercice précédent pour n variant de 2 à 20 par pas de 2 et les $n + 1$ points x_i uniformément répartis entre 0 et 1. Représenter le logarithme du conditionnement de la matrice A en fonction de n. Commenter.

La solution de cet exercice se trouve page 70.

Exercice 3.3 Calculs dans la base de Lagrange

> Pour $n \in \{5, 10, 20\}$, on définit, pour $i = 0, \ldots, n$, les points $x_i = i/n$. Écrire un programme qui, étant donnés les entiers n et k, calcule, par la fonction `polyfit`, les coefficients `cf` du polynôme de Lagrange ℓ_i. Calculer la valeur de $\ell_{[n/2]}$ au point 0. Commenter.

La solution de cet exercice se trouve page 72.

Considérons maintenant la Base 3. On appelle différence divisée d'ordre k de la fonction f, relativement aux $k+1$ points distincts $x_0, \ldots, x_k$ et on note $f[x_0, \ldots, x_k]$ le réel défini pour $k = 0$ par $f[x_i] = f(x_i)$ et pour $k \geqslant 1$ par

$$f[x_0, \ldots, x_k] = \frac{f[x_1, \ldots, x_k] - f[x_0, \ldots, x_{k-1}]}{x_k - x_0}.$$

Le calcul des différences divisées est effectué par l'algorithme, dit de Newton, qui est schématisé par l'arbre suivant :

$$\begin{array}{lllll}
f[x_0] = f(x_0) & & & & \\
& \searrow\nearrow & f[x_0, x_1] = \frac{f(x_1)-f(x_0)}{x_1-x_0} & & \\
f[x_1] = f(x_1) & & & \searrow\nearrow & f[x_0, x_1, x_2] = \frac{f[x_1,x_2]-f[x_0,x_1]}{x_2-x_0} \quad \ldots \\
& \searrow\nearrow & f[x_1, x_2] = \frac{f(x_2)-f(x_1)}{x_2-x_1} & & \vdots \\
f[x_2] = f(x_2) & & \vdots & & \vdots \\
\vdots & & \vdots & & \vdots
\end{array}$$

Dans la première colonne, on écrit les différences divisées d'ordre 0, la deuxième colonne contient les différences divisées d'ordre 1, etc.

La proposition suivante montre que les différences divisées sont les coefficients de $\mathcal{I}_n f$ dans la Base 3 :

Proposition 3.1

$$\mathcal{I}_n f(x) = f[x_0] + \sum_{k=1}^{n} f[x_0, \ldots, x_k](x - x_0)(x - x_1) \cdots (x - x_{k-1}). \qquad (3.5)$$

Notons c un tableau contenant les différences divisées c_i. Pour calculer la valeur du polynôme $\mathcal{I}_n f$ en un point x, on écrit

$$\mathcal{I}_n f(x) = c_0 + (x - x_0)\,\{c_1 + (x - x_1)\,\{c_2 + c_3\,(x - x_2) + \cdots$$

Cette écriture de $\mathcal{I}_n f(x)$ est dite forme de Horner et est très économique puisque le calcul de $\mathcal{I}_n f(x)$ sous la forme de Horner se fait en n multiplications, n soustractions et

n additions alors que le calcul avec l'écriture (3.5) se fait en $n(n+1)/2$ multiplications, $n(n+1)/2$ soustractions et n additions.

```
y = c_n
pour k = n − 1 ↘ 0 faire :
    y = (x − x_k)y + c_k
fin
```

Algorithme de Horner pour calculer $\mathcal{I}_n f(x)$.

Exercice 3.4 Différences divisées

1. Écrire un programme calculant les différences divisées à l'ordre n d'une fonction. On se propose de partir d'un tableau c contenant les $(n+1)$ valeurs $f[x_i] = f(x_i)$. À la première étape, $c_0 = f[x_0]$ est conservé, les autres valeurs c_k $(k \geqslant 1)$ sont remplacées par les différences divisées d'ordre un. À la deuxième étape, $c_1 = f[x_0, x_1]$ est conservé, les valeurs c_k $(k \geqslant 2)$ sont « écrasées », etc.

```
pour k = 0 ↗ n faire :
    c_k ← f(x_k)
fin
pour p = 1 ↗ n faire :
    pour k = n ↘ p faire :
        c_k ← (c_k − c_{k−1})/(x_k − x_{k−p})
    fin
fin
```

Algorithme des différences divisées.

2. Utiliser l'algorithme de Horner pour calculer $\mathcal{I}_n f$ sur une grille fine de points de $[0, 1]$. Tracer $\mathcal{I}_n f$ et f sur la même grille en montrant les points d'interpolation x_i.

La solution de cet exercice se trouve page 72.

➤ Erreur d'interpolation

Étant donné $x \in [a, b]$, on cherche à estimer l'*erreur ponctuelle* ou *locale* :

$$e_n(x) = f(x) - \mathcal{I}_n f(x)$$

qui, bien entendu, est nulle si x coïncide avec l'un des points d'interpolation.

Proposition 3.2 *Si $f \in C^{n+1}([a, b])$, alors pour tout $x \in [a, b]$, il existe $\xi_x \in [a, b]$ tel que*

$$e_n(x) = \frac{1}{(n+1)!}\Pi_n(x) f^{(n+1)}(\xi_x), \tag{3.6}$$

où $\Pi_n(x) = \prod_{i=0}^{n}(x - x_i)$.

➤ Comment choisir les points d'interpolation ?

Pour tout $x \in [a, b]$, on déduit de l'égalité (3.6) la majoration

$$|e_n(x)| \leqslant \frac{1}{(n+1)!} \|\Pi_n\|_\infty \, \|f^{(n+1)}\|_\infty.$$

On va donc chercher à choisir les points d'interpolation de manière à minimiser $\|\Pi_n\|_\infty$, puisque le terme $\|f^{(n+1)}\|_\infty$ ne dépend que de la fonction elle-même et pas des points d'interpolation. Commençons par la situation assez naturelle où les points d'interpolation sont uniformément répartis (on dit aussi équidistants) dans l'intervalle $[a, b]$:

$$x_i = a + i\frac{b-a}{n}, \qquad 0 \leqslant i \leqslant n.$$

On montre qu'il existe dans ce cas une constante c indépendante de n telle que, pour n assez grand

$$\max_{a \leqslant x \leqslant b} |\Pi_n(x)| \geqslant c(b-a)^{n+1} e^{-n} n^{-5/2}. \tag{3.7}$$

Considérons maintenant le cas des points de Tchebycheff. On appelle ainsi les n zéros du polynôme de Tchebycheff T_n. Ce polynôme est défini sur l'intervalle $[-1, 1]$ par

$$T_n(t) = \cos n\theta, \text{ où } \cos\theta = t. \tag{3.8}$$

Les points de Tchebycheff sont donc

$$t_i = \cos(\theta_i), \quad \theta_i = \frac{\pi}{2n} + i\frac{\pi}{n} \qquad 0 \leqslant i \leqslant n-1.$$

Sur un intervalle $[a, b]$, les points de Tchebycheff sont définis comme étant les images des points précédents par la transformation affine φ qui envoie $[-1, 1]$ sur $[a, b]$ (voir la figure 3.2)

$$x_i = \varphi(t_i) = \frac{a+b}{2} + \frac{b-a}{2} cos(\theta_i), \qquad 0 \leqslant i \leqslant n-1.$$

Quels que soient les points x_i' choisis dans $[a, b]$, on peut montrer que

$$\max_{x \in [a,b]} |\prod_{i=0}^{n-1}(x - x_i')| \geqslant \max_{x \in [a,b]} \left|\prod_{i=0}^{n-1}(x - x_i)\right| = \frac{(b-a)^n}{2^{2n-1}}. \tag{3.9}$$

La comparaison des minorations (3.7) et (3.9) est nettement à l'avantage des points de Tchebycheff. Nous verrons au paragraphe suivant une autre raison de préférer ces points à des points uniformément répartis.

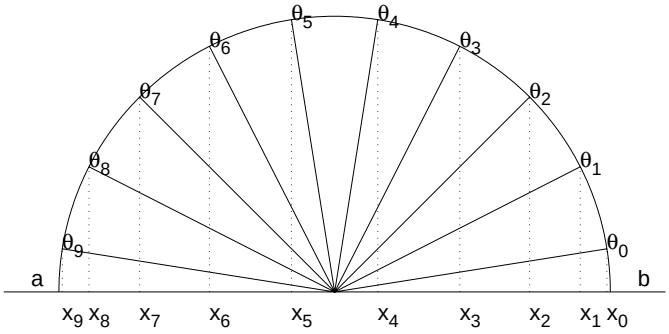


Figure 3.2 Points de Tchebycheff sur $[a, b]$, $n = 10$.

➤ Stabilité de l'interpolation de Lagrange.

On appelle $n^{\text{ème}}$ constante de Lebesgue associée aux points $(x_i^n)_{i=0}^n$ le réel Λ_n défini par

$$\Lambda_n = \max_{x\in[a,b]} \sum_{i=0}^{n} |\ell_i(x)|. \tag{3.10}$$

Il est à noter que Λ_n ne dépend pas de la fonction f, mais uniquement des points x_i. Supposons commettre une erreur ε_i sur chaque valeur $f(x_i)$ et soit $\tilde{p}_n$ le polynôme interpolant les valeurs $\tilde{f}_i = f_i + \varepsilon_i$. L'erreur commise en un point x est $\mathcal{I}_n f - \tilde{p}_n(x) = -\sum_{i=0}^n \varepsilon_i \ell_i(x)$. En posant $\varepsilon = \max_i |\varepsilon_i|$, on obtient la majoration

$$\|\mathcal{I}_n f - \tilde{p}_n\|_\infty \leqslant \varepsilon \Lambda_n,$$

qui montre que la constante Λ_n mesure le facteur d'amplification de l'erreur dans le procédé d'interpolation de Lagrange.

Proposition 3.3 *Quels que soient les points d'interpolation, on a toujours*

$$\lim_{n\to+\infty} \Lambda_n = +\infty. \tag{3.11}$$

On voit ainsi que de petites perturbations sur les données (ε petit) peuvent engendrer de grandes variations sur la solution ($\mathcal{I}_n f$). C'est la situation typique d'un problème *mal conditionné.*

Exercice 3.5 Calcul de la constante de Lebesgue

1. Écrire une fonction calculant la constante de Lebesgue associée à un tableau x de n réels (voir (3.10)). On utilisera les fonction `polyval` et `polyfit` pour calculer les ℓ_i. Le maximum dans (3.10) sera calculé sur une grille uniforme de 100 points pris entre $\min_i x_i$ et $\max_i x_i$.

2. Cas uniforme. Calculer pour n variant de 10 à 30 par pas de 5, la constante de Lebesgue $\Lambda_U(n)$ associée en $n+1$ points uniformément répartis dans l'intervalle $[-1,1]$. Tracer la courbe $n \mapsto \ln(\Lambda_U((n))$. Commenter.
3. Cas des points de Tchebycheff. Calculer pour n variant de 10 à 20 par pas de 5, la constante de Lebesgue $\Lambda_T(n)$ associée aux $n+1$ points de Tchebycheff sur $[-1,1]$. Tracer la courbe $\ln n \mapsto \Lambda_T(n)$. Commenter.

La solution de cet exercice se trouve page 73.

➤ Convergence de l'interpolation de Lagrange.

On s'intéresse maintenant à l'erreur globale $\|e_n\|_\infty = \max_{x\in[a,b]} |e_n(x)|$.

Proposition 3.4 *Pour toute fonction f continue sur $[a,b]$, on a*

$$\|e_n\|_\infty \leqslant (1+\Lambda_n)E_n(f),$$

où $E_n(f) = \inf_{q\in\mathbb{P}_n} \|f-q\|_\infty$ est l'erreur de meilleure approximation de la fonction f par des polynômes de $\mathbb{P}_n$, au sens de la norme de la convergence uniforme.

Remarque 3.1 : L'erreur globale $\|f-\mathcal{I}_n f\|_\infty$ est donc majorée par le produit de deux facteurs. L'un Λ_n tend toujours vers $+\infty$, l'autre $E_n(f)$, converge d'autant plus vite vers 0 que la fonction f est régulière. Il y aura convergence uniforme de l'interpolation de Lagrange si le produit $\Lambda_n E_n(f)$ tend vers 0, c'est-à-dire si la décroissance de $E_n(f)$ compense la croissance de Λ_n.

Exercice 3.6

Calculer et représenter graphiquement (sur une grille uniforme de 100 points), l'interpolée de la fonction $f_1 : x \mapsto |\sin(\pi x)|$ aux n points de Tchebycheff de l'intervalle $[-1,1]$. On prendra $n = 20$, 30 puis 40. Mêmes questions pour la fonction $f_2 : x \mapsto x f_1(x)$. Commenter les résultats.

La solution de cet exercice se trouve page 75.

Exercice 3.7 Phénomène de Runge

Calculer et représenter graphiquement sur une grille uniforme de 100 points, l'interpolée de la fonction $f : x \mapsto 1/(x^2+a^2)$ aux $n+1$ points $x_i = -1+2i/n$ ($i = 0, \cdots, n$). On prendra $a = 2/5$ et $n = 5$, 10 puis 15. On notera que la fonction à interpoler est très régulière (de classe $\mathcal{C}^\infty$ sur $\mathbb{R}$), ce qui n'est pas le cas des fonctions considérées à l'exercice précédent. Commenter les résultats.

La solution de cet exercice se trouve page 76.

b) Interpolation de Hermite

On suppose que la fonction f a des dérivées d'ordre α_i en chaque point x_i, il existe alors un unique polynôme $p_n \in \mathbb{P}_n$ tel que pour tous $i = 0, \ldots, k$ et $j = 0, \ldots, \alpha_i$, on ait

$$p_n^{(j)}(x_i) = f^{(j)}(x_i). \tag{3.12}$$

Le polynôme p_n qu'on notera $\mathcal{I}_n(f; x_0, \ldots, x_k; \alpha_0, \ldots, \alpha_k)$, ou plus simplement $\mathcal{I}_n^H f$ est appelé polynôme d'interpolation de f au sens d'Hermite aux points x_i, relativement aux indices α_i.

Théorème 3.2 *Si $f \in \mathcal{C}^{n+1}([a,b])$, alors pour tout $x \in [a,b]$, il existe $\xi_x \in [\min_i x_i, \max_i x_i]$ tel que*

$$e_n^H(x) = f(x) - \mathcal{I}_n^H f(x) = \frac{1}{(n+1)!}\Pi_n^H(x) f^{(n+1)}(\xi_x), \tag{3.13}$$

où $\Pi_n^H(x) = \prod_{i=0}^{k} (x - x_i)^{1+\alpha_i}$.

On suppose la fonction f de classe $\mathcal{C}^{n+1}$ sur l'intervalle $[a,b]$. Quels que soient les points distincts $x_0, \cdots, x_n$, $n+1$ points de l'intervalle $[a,b]$, il existe $\xi \in [a,b]$ tel que

$$f^{(n)}(\xi) = n! f[x_0, \cdots, x_n].$$

Cette relation définit un lien entre les différences divisées et les dérivées. Plus précisément

Remarque 3.2 : En faisant tendre chaque x_i vers x, on obtient une approximation de la dérivée $n^{\text{ième}}$ de f au point x :

$$\lim_{x_i \to x} f[x_0, \cdots, x_n] = \frac{1}{n!} f^{(n)}(x).$$

On peut utiliser l'algorithme de Newton pour calculer le polynôme d'interpolation au sens d'Hermite d'une fonction, utilisant la remarque 3.2 comme dans l'exemple suivant.

Exemple 1 : Calculer le polynôme d'interpolation p de degré minimal vérifiant

$$p(0) = -1,\ p(1) = 0,\ p'(1) = \alpha \in \mathbb{R} \text{ donné.} \tag{3.14}$$

On écrit

$x_0 = 0 \quad f[x_0] = \boxed{-1}$

$\searrow \nearrow \quad f[x_0, x_1] = \boxed{1}$

$x_1 = 1 \quad f[x_1] = 0 \qquad \searrow \nearrow \quad f[x_0, x_1, x_2] = \frac{\alpha - 1}{1 - 0} = \boxed{\alpha - 1}$

$\searrow \nearrow \quad \boxed{\boxed{f'(1)}} = \alpha$

$x_2 = 1 \quad f[x_1] = 0$

On trouve

$$p(x) = \boxed{-1} + \boxed{1}x + \boxed{(\alpha - 1)}x(x - 1).$$

On a écrit en fait $x_2 = 1 + \varepsilon$, $f[x_1, x_2] = (f(1+\varepsilon) - f(1))/(1 + \varepsilon - 1)$, puis utilisé le fait que quand ε tend vers 0, $f[x_1, x_2]$ tend vers $f'(1)$.

Exercice 3.8

On note $f : x \mapsto e^{-x}\cos(3\pi x)$.

1. Écrire une fonction basée sur les différences divisées (comme à l'exemple 1) calculant le polynôme interpolant une fonction f au sens de Hermite (incluant Lagrange). On donnera en argument d'entrée de cette fonction les points (distincts) d'interpolation x_i et pour chaque point x_i, la dérivée maximale α_i interpolée en ce point ainsi que les valeurs $f^{(\ell)}(x_i)$ pour $\ell = 0, \cdots, \alpha_i$.
2. Calculer l'interpolée de Lagrange de la fonction f aux points $0, 1/4, 3/4$ et 1. Représenter f et son polynôme d'interpolation sur l'intervalle $[0, 1]$.
3. Calculer l'interpolée au sens d'Hermite de la fonction f aux mêmes points, on interpolera f et f'. Représenter f et le nouveau polynôme d'interpolation sur l'intervalle $[0, 1]$. Comparer avec les résultats précédents.
4. Mêmes questions en interpolant cette fois f et f' aux points précédents et au point $1/2$.

La solution de cet exercice se trouve page 76.

Exercice 3.9

Tracer, sur $[0, 1]$, et pour diverses valeurs de m, le polynôme de degré minimal p tel que

$$p(0) = 0, p(1) = 1, \text{ et } p^{(\ell)}(0) = p^{(\ell)}(1) = 0, \text{ pour } \ell = 1, \ldots, m.$$

La solution de cet exercice se trouve page 77.

3.2.2 Meilleure approximation polynomiale

On cherche cette fois un polynôme qui est le « plus proche » de f au sens d'une certaine norme $\|.\|_{\mathcal{X}}$, $\mathcal{X}$ étant un espace vectoriel normé de fonctions, contenant les polynômes. Pour $f \in \mathcal{X}$, on appelle meilleure approximation polynomiale de f dans $\mathbb{P}_n$, un polynôme $p_n \in \mathbb{P}_n$ vérifiant

$$\|f - p_n\|_{\mathcal{X}} = \inf_{q \in \mathbb{P}_n} \|f - q\|_{\mathcal{X}}. \tag{3.15}$$

La quantité $\inf_{q\in\mathbb{P}_n} \|f - q\|_{\mathcal{X}}$ est appelée erreur de meilleure approximation de f dans $\mathbb{P}_n$, au sens de la norme $\|\,.\,\|_{\mathcal{X}}$. Nous allons étudier le problème de la meilleure approximation dans les deux cas suivants.

– *Cas 1.* $I = [a, b], \mathcal{X} = \mathcal{C}(I)$, ensemble des fonctions continues sur I, muni de la norme de la convergence uniforme que nous noterons $\|.\|_\infty$, On notera $E_n(f)$ l'erreur de meilleure approximation uniforme.

– *Cas 2.* $I =]a, b[, \mathcal{X} = L^2(I)$, ensemble des fonctions mesurables, définies sur I et telles que $\int_a^b |f(x)|^2 dx < +\infty$. $L^2(I)$ est muni du produit scalaire et de la norme

$$\langle f, g\rangle = \int_{-1}^{1} f(t)g(t)dt, \quad \|f\| = \sqrt{\langle f, f\rangle}.$$

a) Meilleure approximation uniforme

Dans cette partie $I = [a, b]$ et $f \in \mathcal{X} = \mathcal{C}(I)$. On cherche $p_n \in \mathbb{P}_n$ solution du problème

$$\|f - p_n\|_\infty = \inf_{q\in\mathbb{P}_n} \|f - q\|_\infty = (E_n(f)).$$

La notion suivante permet de caractériser le polynôme de meilleure approximation uniforme d'une fonction.

Définition 3.1 *On dit qu'une fonction continue φ est équioscillante sur $(n+1)$ points d'un intervalle réel $[a, b]$ s'il existe $(n + 1)$ points $x_0 < x_1 < \cdots < x_n$ de $[a, b]$ en lesquels φ prend alternativement les valeurs $\pm\|\varphi\|_\infty$ (voir la figure 3.3).*

Proposition 3.5 *Soit f une fonction continue sur $I = [a, b]$, le polynôme p_n de meilleure approximation uniforme de f dans $\mathbb{P}_n$ est le seul polynôme de $\mathbb{P}_n$ pour lequel la fonction $f - p_n$ équioscille en (au moins) $n + 2$ points distincts de I.*

Par exemple, la meilleure approximation uniforme d'une fonction f continue sur $[a, b]$ par des constantes est réalisée par $p_0 = \frac{1}{2}\{\min_{x\in[a,b]} f(x) + \max_{x\in[a,b]} f(x)\}$ et il existe (au moins) deux points où la fonction $f - p_0$ équioscille, ce sont les points où la fonction continue f atteint ses extrema sur le compact $[a, b]$.

Pour déterminer la meilleure approximation uniforme d'une fonction f, il suffit donc de trouver un polynôme $p \in \mathbb{P}_n$ et $n + 2$ points tels que $f - p$ équioscille en ces

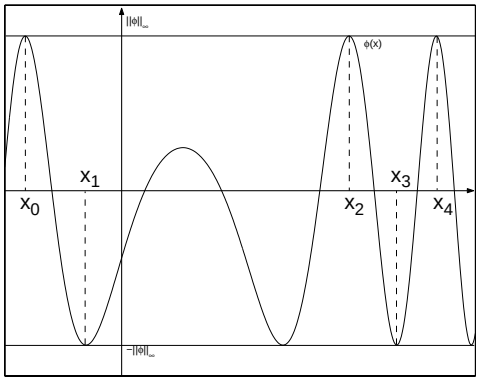


Figure 3.3 Exemple de fonction équioscillante.

points. C'est ce que réalise l'algorithme suivant.

Initialisation. On choisit $n+2$ points distincts $x_0^0 < x_1^0 < \cdots < x_{n+1}^0$.
Etape k. Supposons connus $n+2$ points $x_0^k < x_1^k < \cdots < x_{n+1}^k$.
On construit (voir l'exercice 3.10) un polynôme $p_k \in \mathbb{P}_n$ tel que
$f(x_i^k) - p_k(x_i^k) = (-1)^i \{f(x_0^k) - p_k(x_0^k)\}, \qquad i = 1, \cdots, n+1$
(*a*) Si $\|f - p_k\|_\infty = |f(x_i^k) - p_k(x_i^k)|$ (*i*)
l'algorithme s'arrête car la fonction $f - p_k$ équioscille en ces points et donc p_k est le polynôme de meilleure approximation uniforme de f.
(*b*) Sinon, il existe $y \in [a, b]$ vérifiant
$$\|f - p_k\|_\infty = |f(y) - p_k(y)| \; > \; |f(x_i^k) - p_k(x_i^k)|, \quad i = 0, \cdots, n+1. \qquad (ii)$$
On construit alors un nouvel ensemble de points
$$x_0^{k+1} < x_1^{k+1} < \cdots < x_{n+1}^{k+1}$$
en remplaçant un des points x_i^k par y de telle sorte que
$$\left(f(x_j^{k+1}) - p_k(x_j^{k+1})\right)\left(f(x_{j-1}^{k+1}) - p_k(x_{j-1}^{k+1})\right) \leqslant 0, \quad j = 1, \cdots, n+1.$$

Algorithme de Remez.

Exercice 3.10

Montrer qu'il existe un polynôme et un seul $p_k \in \mathbb{P}_n$ défini à l'étape k de l'algorithme de Remez. Programmer une fonction qui calcule ce polynôme, à partir de la donnée de $n+2$ points x_i et d'une fonction f. On pourra écrire $p(t) = \sum_{j=0}^{n} p_j t^j$ et résoudre par MATLAB un système linéaire dont le vecteur $(p_0, \cdots, p_n)^T$ est solution.

La solution de cet exercice se trouve page 78.

Exercice 3.11 Algorithme de Remez

On se propose de calculer la meilleure approximation uniforme de la fonction $x \mapsto \sin(2\pi\cos(\pi x))$ sur $[0,1]$ par l'algorithme de Remez. Pour assurer les inégalités, on considérera les divers cas :

$$y \in]x_i^k, x_{i+1}^k[\quad \text{et} \quad (f(x_i^k) - p_k(x_i^k))(f(y) - p_k(y)) \geqslant 0$$

ou $(f(x_i^k) - p_k(x_i^k))(f(y) - p_k(y)) < 0$. Dans le premier cas x_{i+1} est remplacé par y, dans l'autre cas, c'est x_i qui est remplacé par y. Considérer aussi le cas ou $y < \min_i x_i$ et le cas $y > \max_i x_i$. D'autre part l'inégalité (*ii*) de l'algorithme de Remez sera interprétée comme suit :

- la norme $\|f - p_k\|_\infty$ est calculée sur une grille uniforme de 100 points de l'intervalle $[0,1]$;
- l'égalité (*i*) de l'algorithme de Remez est considérée comme vraie si la valeur absolue de la différence entre les deux quantités est plus petite qu'une « tolérance » fixée, par exemple à 10^{-8}.

On comparera les résultats en notant le nombre d'itérations de l'algorithme de Remez avant convergence pour trois choix des $n+2$ points x_i à l'initialisation.

- Points équidistants : $x_i = \frac{i}{n+1}$, $i = 0, \cdots, n+1$.
- Points de Tchebycheff : $x_i = \frac{1}{2}(1 - cos(i\frac{\pi}{n+1})$, $i = 0, \cdots, n+1$.
- Points choisis aléatoirement : les x_i sont $n+1$ points obtenus par la fonction `rand`, puis ordonnés.

La solution de cet exercice se trouve page 79.

b) Meilleure approximation hilbertienne

Par transformation affine, on peut toujours ramener l'intervalle $]a,b[$ à l'intervalle $]-1,1[$. La structure hilbertienne de l'espace $\mathcal{X} = L^2(I)$ permet d'étendre à cet espace, de dimension infinie, des concepts habituels en dimension finie : base, projection orthogonale, ... Voir par exemple [Schwartz, 1980] pour les définitions et résultats de cette section.

Le problème qui nous intéresse ici est de déterminer la meilleure approximation d'une fonction de $L^2(I)$ par des polynômes d'un degré donné.

c) Base hilbertienne

Les polynômes de Legendre sont définis par la relation de récurrence

$$(n+1)L_{n+1}(x) = (2n+1)xL_n(x) - nL_{n-1}(x) \qquad (n \geqslant 1)$$

avec $L_0(x) = 1$ et $L_1(x) = x$. Le degré de L_n est égal à n et pour tout couple d'entiers (n, m)

$$\langle L_n, L_m \rangle = \begin{cases} 0 & \text{si } n \neq m \\ 1/(n+1/2) & \text{si } n = m. \end{cases}$$

On dit que ces polynômes sont orthogonaux. On présente sur la figure 3.4 quelques polynômes de Legendre.

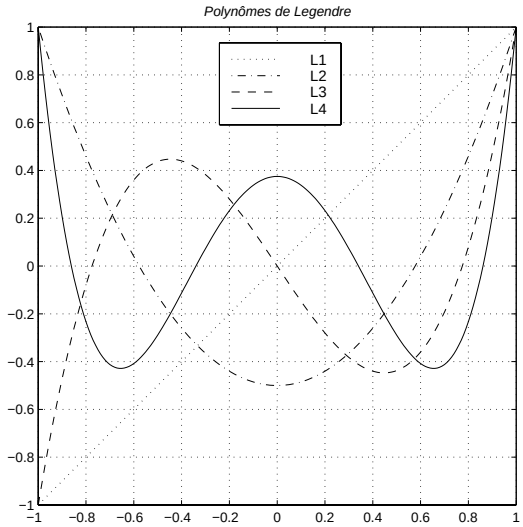


Figure 3.4 Exemple de polynômes orthogonaux : les polynômes de Legendre.

La famille $L_n^* = L_n/\|L_n\|$ forme une base hilbertienne de $L^2(I)$, c'est-à-dire que la famille $(L_n^*)_{n\geqslant 0}$ est orthonormée et l'ensemble des combinaisons linéaires finies des L_n^* est dense dans $L^2(I)$. Comme en dimension finie, on peut développer toute fonction de $L^2(I)$ dans la base (infinie) des polynômes de Legendre.

Théorème 3.3 *Soit $f \in L^2(I)$ et $n \in \mathbb{N}$.*

1. *f est développable en série de Legendre : c'est-à-dire qu'il existe des réels $\hat{f}_k$ tels que*

$$f = \sum_{k=0}^{\infty} \hat{f}_k L_k. \tag{3.16}$$

2. *Il existe un unique polynôme de $\mathbb{P}_n$ (qu'on notera $\pi_n f$) réalisant la meilleure approximation polynomiale hilbertienne de f dans $\mathbb{P}_n$, c'est-à-dire*

$$\|f - \pi_n f\| = \inf_{q\in\mathbb{P}_n} \|f - q\|.$$

De plus $\pi_n f$ est caractérisé par les relations d'orthogonalité (voir figure 3.5)

$$\langle f - \pi_n f, p\rangle = 0, \qquad \forall p \in \mathbb{P}_n \tag{3.17}$$

qui signifient que $\pi_n f$ est la projection orthogonale de f sur $\mathbb{P}_n$.

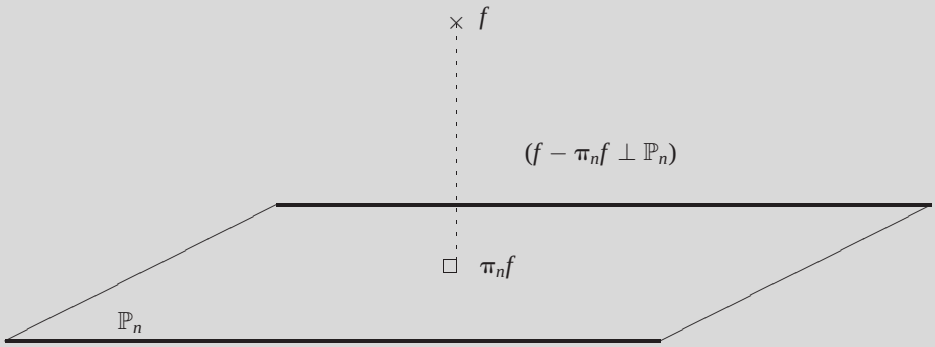


Figure 3.5 La meilleure approximation hilbertienne de f est sa projection orthogonale sur $\mathbb{P}_n$.

L'existence et l'unicité de $\pi_n f$ résulte du fait que $\mathbb{P}_n$ est un sous-espace vectoriel de dimension finie de l'espace de Hilbert $L^2(I)$. Les réels $\hat{f}_k$ dans (3.16) sont appelés les coefficients de Legendre (ou de Fourier-Legendre) de la fonction f. On déduit de l'orthogonalité des polynômes de Legendre que

$$\hat{f}_k = \frac{\langle f, L_k\rangle}{\|L_k\|^2} = (k + 1/2)\int_{-1}^{1} f(t)L_k(t)dt. \tag{3.18}$$

d) Calcul de la meilleure approximation

On développe $\pi_n f \in \mathbb{P}_n$ dans la base des $(L_k)_{k=0}^n$, $\pi_n f = \sum_{k=0}^n \alpha_k L_k$. Pour déterminer les coefficients α_k, on se sert des égalités (3.17)

$$(\forall i = 0, \ldots, n) \quad \langle f, L_i\rangle = \langle \pi_n f, L_i\rangle = \sum_{i=0}^{n} \alpha_k \langle L_k, L_i\rangle = \alpha_i \|L_i\|^2.$$

Les α_i sont donc les coefficients de Legendre de f et $\pi_n f$ est la série de f, tronquée à l'ordre n :

$$\pi_n f = \sum_{k=0}^{n} \hat{f}_k L_k. \tag{3.19}$$

Le calcul de la meilleure approximation d'une fonction se ramène donc au calcul des coefficients de Legendre de cette fonction. Comme l'intégrale dans (3.18) ne peut (en général) pas être calculée exactement, on a recours à des techniques d'intégration

numérique (on dit aussi quadrature), voir à ce sujet [Crouzeix et Mignot, 1989] et [Delabrière et Postel, 2004].

Le résultat suivant montre la convergence de la meilleure approximation.

Proposition 3.6 *Pour tout $f \in L^2(I)$, on a*

$$\lim_{n\to+\infty} \|f - \pi_n f\| = 0. \tag{3.20}$$

Les polynômes de Legendre seront utilisés au chapitre 5 pour résoudre une équation différentielle.

e) Approximation au sens des moindres carrés discrets

Dans cette section, on cherche toujours à approcher une fonction f par un polynôme, mais au sens d'une norme discrète. Étant donnés m points $(x_i)_{i=1}^m$ distincts deux à deux et m valeurs $(y_i)_{i=1}^m$, on cherche à déterminer un polynôme $p = \sum_{j=0}^{n-1} a_j x^j \in \mathbb{P}_{n-1}$ qui minimise la quantité

$$E = \sum_{i=1}^{m} |y_i - p(x_i)|^2, \tag{3.21}$$

où m est en général très grand par rapport à n. En termes géométriques, on cherche à faire passer la courbe représentative de p le plus près possible (au sens de la norme euclidienne) du nuage de points (x_i, y_i). La fonction E définie par (3.21) est une fonction des n variables $(a_0, a_1, \cdots, a_{n-1})$. Cette fonction admet un minimum en l'unique point de $\mathbb{R}^n$ qui annule ses dérivées partielles. Comme

$$\frac{\partial E}{\partial a_j} = 0 \iff -2\sum_{i=1}^{m} (y_i - p(x_i))\, x_i^j = 0 \iff \sum_{k=0}^{n-1} \left(\sum_{i=1}^{m} x_i^{k+j} \right) a_k = \sum_{i=1}^{m} x_i^j y_i,$$

on peut calculer le point $a = (a_0, \ldots, a_{n-1})^T$ où la fonction f prend sa valeur minimale en résolvant le système linéaire

$$\widetilde{A} a = \widetilde{b}, \tag{3.22}$$

dont la matrice et le second membre sont

$$\widetilde{A} = \begin{pmatrix} \sum_i 1 & \sum_i x_i & \cdots & \sum_i x_i^{n-1} \\ \sum_i x_i & \sum_i x_i^2 & \cdots & \sum_i x_i^n \\ \vdots & \vdots & \vdots & \vdots \\ \sum_i x_i^{n-1} & \sum_i x_i^n & \cdots & \sum_i x_i^{2n-1} \end{pmatrix} \in \mathbb{R}^{n\times n}, \quad \widetilde{b} = \begin{pmatrix} \sum_i y_i \\ \sum_i x_i y_i \\ \vdots \\ \sum_i x_i^{n-1} y_i \end{pmatrix}.$$

Étudions d'abord le cas $n = 2$ où on cherche à déterminer une droite (dite *des moindres carrés*). Dans ce cas la matrice $\tilde{A}$ et le vecteur $\widetilde{b}$ sont

$$\widetilde{A} = \begin{pmatrix} m & \sum_i x_i \\ \sum_i x_i & \sum_i x_i^2 \end{pmatrix}, \quad \widetilde{b} = \begin{pmatrix} \sum_i y_i \\ \sum_i x_i y_i \end{pmatrix}. \tag{3.23}$$

Le déterminant de $\widetilde{A}$

$$\Delta = m(\sum_{i=1}^{m} x_i^2) - (\sum_{i=1}^{m} x_i)^2 = m\sum_{i=1}^{m}(x_i - \frac{1}{m}\sum_{j=1}^{m} x_j)^2$$

n'est nul que dans le cas où tous les points x_i sont identiques. Comme on a supposé ces points distincts, la matrice $\widetilde{A}$ est inversible et le système (3.23) a une solution unique.
Revenons au cas général. Soit A la matrice de Van der Monde définie par

$$A = \begin{pmatrix} 1 & x_1 & \dots & x_1^{n-1} \\ \vdots & \vdots & & \vdots \\ 1 & x_m & \dots & x_m^{n-1} \end{pmatrix} \in \mathbb{R}^{m\times n}.$$

Notant que $\tilde{A} = A^T A$ et $\tilde{b} = A^T b$ avec $b = (y_1, \dots, y_m)^T$, le système (3.22) s'écrit

$$A^T A a = A^T b. \tag{3.24}$$

Le résultat suivant montre que les solutions de (3.24) sont les solutions du problème de minimisation : trouver $a \in \mathbb{R}^n$ tel que

$$\|Aa - b\| = \inf_{x\in\mathbb{R}^n} \|Ax - b\|, \tag{3.25}$$

Théorème 3.4 *$a \in \mathbb{R}^n$ est solution des équations normales (3.24) si et seulement si a est solution du problème de minimisation (3.25).*

On peut donc pour résoudre le problème des moindres carrés, soit résoudre le problème (3.25) par des algorithmes d'optimisation (recherche du minimum d'une fonction), soit résoudre le problème (3.24) par des techniques de résolution de systèmes linéaires (voir par exemple [Allaire et Kaber, 2002 (1)] ou [Joly, 2004]).

Pour calculer la meilleure approximation au sens des moindres carrés par MATLAB , on n'aura pas besoin de définir la matrice A, l'appel `polyfit(x,y,n)` où le vecteur x est un vecteur de valeurs x_i et y un vecteur contenant les valeurs y_i renvoie les coefficients d'un polynôme de degré n solution du problème des moindres carrés.

Exercice 3.12

On se propose de calculer la meilleure approximation au sens des moindres carrés de la fonction f de l'exercice 3.11. Le degré n « optimal" du polynôme de meilleure approximation p_n est déterminé de la façon suivante. Partant de $n = 0$, on incrémente n de 1 jusqu'à ce que l'erreur relative entre deux solutions $|e_n - e_{n-1}|/e_{n-1}$ soit plus petite que $1/2$, par exemple. On a noté ici $e_n = \|x - p_n(x)\|_2$.

La solution de cet exercice se trouve page 80.

3.3 APPROXIMATION POLYNOMIALE PAR MORCEAUX

La figure 3.6 représente les interpolations polynomiales de la fonction f définie sur $[0, 1]$ par

$$f(x) = \begin{cases} 1 & \text{si } 0 \leqslant x \leqslant 0.25 \\ 2 - 4x & \text{si } 0.25 \leqslant x \leqslant 0.5 \\ 0 & \text{si } 0.5 \leqslant x \leqslant 1 \end{cases}$$

sur l'intervalle $[0, 1]$ en respectivement 4, 6, 8 et 10 points. Visiblement, il y a un problème dû au manque de régularité de f sur la *totalité* de l'intervalle $I = [0, 1]$. Cette fonction a pourtant une structure simple ; elle est affine sur chacun des intervalles $[0, 1/4]$, $[1/4, 1/2]$ et $[1/2, 1]$.

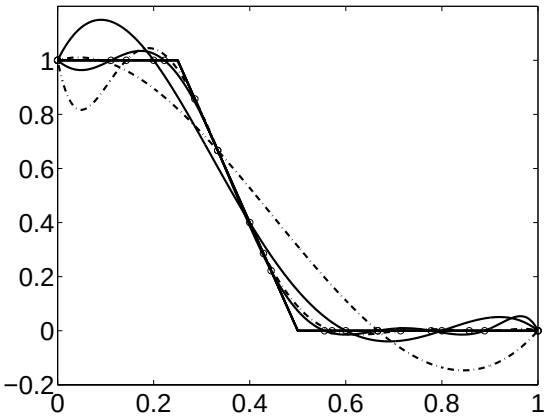


Figure 3.6 Interpolation d'une fonction affine par morceaux.

Soit f une fonction continue, définie sur l'intervalle $[0, 1]$, on cherche à approcher f par une autre fonction S qui soit polynomiale par morceaux, une telle fonction est appelée *spline*. L'intérêt de l'approximation polynomiale par morceaux réside dans la possibilité de mieux contrôler les problèmes liés au manque de régularité globale de la fonction, un autre intérêt pratique est la stabilité dans les calculs : on commet moins d'erreurs en calculant plusieurs polynômes de bas degré qu'en calculant un seul polynôme de degré élevé, à précision de calcul égale.

On découpe l'intervalle $I = [0, 1]$ en sous-intervalles I_i de même longueur, $I_i = [x_i, x_{i+1}]$, $x_i = ih$, $h = 1/n$. Sur chaque sous-intervalle I_i on approche f par un polynôme $p_{n,i}$ de degré n. On note S_n la fonction polynomiale par morceaux coïncidant avec $p_{n,i}$ sur l'intervalle I_i.

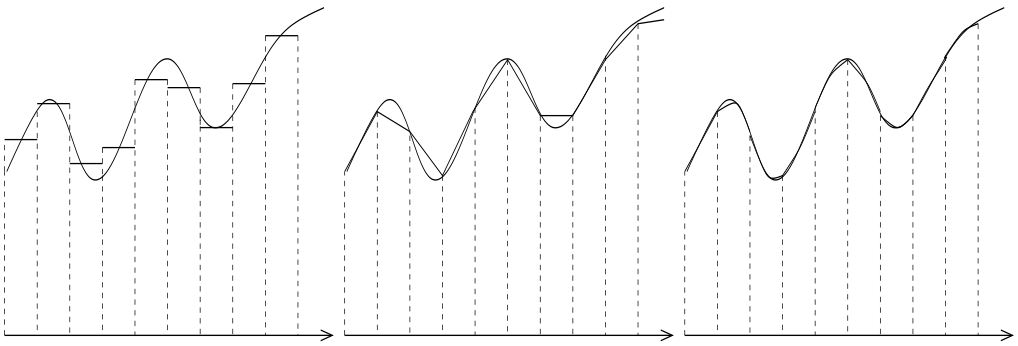


Figure 3.7 De gauche à droite : Exemples d'approximations constante, affine et cubique par morceaux.

3.3.1 Approximation constante par morceaux

Soit S_0 une fonction constante sur chaque intervalle I_i et coïncidant avec f aux points $x_{i+1/2} = (x_i + x_{i+1})/2$

$$S_{0|I_i}(x) = f(x_{i+1/2}).$$

Supposons la fonction f de classe $\mathcal{C}^1$ sur I. D'après la proposition 3.2, pour tout $x \in I_i$, il existe $\xi_{x,i} \in I_i$ tel que

$$f(x) - S_0(x) = (x - x_{i+1/2})f'(\xi_{x,i}).$$

On en déduit, en notant M_1 un majorant de f' sur I, que

$$\|f - S_0\|_\infty \leqslant \frac{h}{2} M_1 \tag{3.26}$$

et donc que S_0 converge uniformément vers f.

> **Remarque 3.3 :** La puissance de h dans la majoration (3.26) nous indique que quand le pas de dicrétisation h est divisé par une constante $c > 0$, il faut s'attendre à ce que l'erreur $\|f - S_0\|_\infty$ soit aussi divisée par cette même constante c.

Exercice 3.13

On note $f : [0, 1] \mapsto f(x) = \sin(4\pi x)$. Représenter la courbe $\ln n \mapsto \ln \|f - S_0\|_\infty$ et retrouver (une approximation de) la majoration (3.26). On pourra prendre les valeurs $n = 10\,k$ avec $k = 1, \ldots, 10$.

La solution de cet exercice se trouve page 80.

3.3.2 Approximation affine par morceaux

Cette fois, l'approximation S_1 est affine sur chaque intervalle I_i et coïncide avec f aux points x_i et x_{i+1} :

$$S_{1|I_i}(x) = \frac{f(x_{i+1}) - f(x_i)}{h}(x - x_i) + f(x_i).$$

Supposons d'abord que la fonction f soit de classe $\mathcal{C}^2$ sur I. D'après la proposition 3.2, pour tout $x \in I_i$, il existe alors $\xi_{x,i} \in I_i$ tel que

$$f(x) - S_1(x) = \frac{(x - x_i)(x - x_{i+1})}{2} f''(\xi_{x,i}).$$

On en déduit, en notant M_2 un majorant de f'' sur I, que

$$\|f - S_1\|_\infty \leqslant \frac{h^2}{8} M_2 \tag{3.27}$$

puis que S_1 converge uniformément vers f.

> **Remarque 3.4 :** La puissance de h dans la majoration (3.27) nous indique que quand le pas de dicrétisation h est divisé par une constante $c > 0$, il faut s'attendre à ce que l'erreur $\|f - S_0\|_\infty$ soit divisée par c^2. Par exemple, en divisant le pas h par 2, on divise la majoration de l'erreur par 4.

Exercice 3.14

Mêmes questions qu'à l'exercice précédent pour retrouver la majoration (3.27).

La solution de cet exercice se trouve page 81.

Si la fonction f n'est que de classe $\mathcal{C}^1$, on peut démontrer encore la convergence. En notant que

$$f(x) = f(x_i) + \int_{x_i}^{x} f'(t)dt, \quad \text{et } S_1(x) = f(x_i) + \frac{x - x_i}{h} \int_{x_i}^{x_{i+1}} f'(t)dt$$

on obtient la majoration

$$\|f - S_1\|_\infty \leqslant 2hM_1,$$

ce qui implique encore la convergence uniforme de S_1 vers f. On notera cependant une moindre précision dans le contrôle de l'erreur (comparer avec (3.27).

3.3.3 Approximation cubique par morceaux (spline cubique)

Cette fois, l'approximation S_3 est de classe $\mathcal{C}^2$ sur I, cubique sur chaque intervalle I_i et coïncide avec f aux points x_i et x_{i+1}. Soit p_i la restriction de S_3 à l'intervalle I_i. On peut chercher p_i sous la forme $p_i(x) = a_i(x - x_i)^3 + b_i(x - x_i)^2 + c_i(x - x_i) + d_i$. Les fonctions f et p_i coïncidant aux points x_i et x_{i+1}, on a

$$d_i = f(x_i) \tag{3.28}$$

et

$$f(x_{i+1}) = a_i h^3 + b_i h^2 + c_i h + d_i. \tag{3.29}$$

Nous allons exprimer les inconnues a_i, b_i et c_i en fonction des dérivées secondes de la fonction. On note $\alpha_i = p_i''(x_i)$. On écrivant la continuité de la dérivée seconde aux points x_i (*i.e.* $p_{i-1}''(x_i) = p_i''(x_i)$), on obtient

$$b_i = \frac{1}{2}\alpha_i \tag{3.30}$$

$$a_i = \frac{\alpha_{i+1} - \alpha_i}{6h} \tag{3.31}$$

Substituant ces deux relations dans (3.29), on obtient

$$f_{i+1} = \frac{\alpha_{i+1} - \alpha_i}{6} h^2 + \frac{1}{2}\alpha_i h^2 + c_i h + f_i$$

d'où l'expression de c_i

$$c_i = \frac{f_{i+1} - f_i}{h} - \frac{2\alpha_i + \alpha_{i+1}}{6} h. \tag{3.32}$$

En écrivant maintenant la continuité de la dérivée première au point x_i, on obtient

$$3a_{i-1}h^2 + 2b_{i-1}h + c_{i-1} = c_i. \tag{3.33}$$

Remplaçant les a_i, b_i, c_i figurant dans (3.33) par leurs expressions données par (3.30), (3.31), (3.32), on obtient $n-1$ équations qui expriment une relation de récurrence entre α_{i-1}, α_i et α_{i+1} :

$$h(\alpha_{i-1} + 4\alpha_i + \alpha_{i+1}) = \frac{6}{h}(f_{i-1} - 2f_i + f_{i+1}).$$

À ces $n-1$ équations, il faut en ajouter deux autres pour fermer le système et déterminer les $n+1$ valeurs α_i. On peut par exemple, poser $\alpha_0 = 0$ et $\alpha_n = 0$, on parle alors de spline naturelle. On peut aussi fixer les pentes de la spline aux extrémités de l'intervalle. Dans le premier cas ($\alpha_0 = 0$, $\alpha_n = 0$), le vecteur $\alpha = (\alpha_1, \cdots, \alpha_{n-1})^T$ est solution du système linéaire tridiagonal $Ax = b$, avec

$$A = h \begin{pmatrix} 4 & 1 & 0 & \dots & 0 \\ 1 & 4 & 1 & & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & & 1 & 4 & 1 \\ 0 & \dots & 0 & 1 & 4 \end{pmatrix} \quad \text{et} \quad b = \frac{6}{h} \begin{pmatrix} f_0 - 2f_1 + f_2 \\ \vdots \\ f_{i-1} - 2f_i + f_{i+1} \\ \vdots \\ f_{n-2} - 2f_{n-1} + f_n \end{pmatrix} \tag{3.34}$$

La matrice A est inversible, car à diagonale strictement dominante. Connaissant les α_i, on détermine les p_i par les relations (3.28), (3.30), (3.31) et (3.32) et la spline est parfaitement déterminée sur chaque intervalle.

Exercice 3.15

Écrire un programme calculant la spline cubique naturelle approchant une fonction donnée (par exemple $f(x) = \sin(4\pi x)$) aux $n + 1$ points $(i/n)_{i=0}^{n}$. On prendra $n = 5$ puis $n = 10$. Tracer la fonction f et la spline sur un même graphique. Il s'agit de voir le comportement de la spline en dehors des points x_i, il sera donc nécessaire de rajouter des points supplémentaires pour représenter les fonctions (prendre dix ou vingt points dans chaque intervalle I_i).

La solution de cet exercice se trouve page 82.

3.4 EN SAVOIR PLUS

Nous n'avons pas traité dans ce chapitre de l'approximation trigonométrique. Tous les résultats énoncés ici ont cependant des analogues dans le cas de fonctions périodiques : existence et unicité d'une meilleure approximation polynomiale, interpolation, fonctions splines, ...

Il existe un outil fort utile dans le calcul de solutions périodiques d'équations différentielles, c'est la transformée de Fourier rapide ou FFT. Cet algorithme permet de calculer facilement (c'est-à-dire en peu d'opérations) les coefficients de Fourier d'une fonction connaissant les valeurs de cette fonction sur une grille de points. Au chapitre 12, l'approximation trigonométrique est utilisée pour résoudre les équations de Navier-Stokes.

L'utilisation des polynômes de Legendre pour résoudre une équation différentielle est proposée au chapitre 5. Pour l'analyse numérique des méthodes spectrales, nous renvoyons le lecteur à [Bernardi, Maday et Rapetti, 2004].

Les ondelettes sont étudiées au chapitre 6. Ces fonctions permettent d'analyser un signal en le localisant, non seulement en fréquence (comme dans l'analyse de Fourier) mais aussi en position, voir à ce sujet [Cohen, 2003].

Les courbes de Bézier sont étudiées au chapitre 9. Il s'agit de construire des courbes paramétrées $(x(t), y(t))$, $t \in I$ définies à partir de « points de contrôle ». En déplaçant les points de contrôle, on déforme les courbes de façon à leur donner la forme souhaitée. Les courbes de Bézier font partie d'un ensemble de techniques utilisées en Conception Graphique Assistée par Ordinateur.

3.5 SOLUTIONS ET PROGRAMMES

Les programmes nécessaires à la résolution des exercices de ce chapitre peuvent être téléchargés à partir du site web du livre.

Solution de l'exercice 3.1

1. $b = (f(x_0), \ldots, f(x_n))^T$ et $A = \begin{pmatrix} 1 & x_0 & x_0^2 & \ldots & x_0^n \\ 1 & x_1 & x_1^2 & \ldots & x_1^n \\ \vdots & \vdots & \vdots & \vdots & \\ 1 & x_n & x_n^2 & \ldots & x_n^n \end{pmatrix}$.
2. `n=10;x=sort(rand(n+1,1));`
3.
```
A=ones(length(x),1);
for k=1:length(x)-1
   A=[A x.^k];
end;
```
4.
```
cf=A\ test1(x);
%remettre dans le "bon" ordre les coefficients
cf=cf(end:-1:1);
y=polyval(cf,x);
plot(x,test1(x),x,y,'r+');
```
La fonction `test1` est définie par
test1=**inline**('**sin**(10.*x.***cos**(x))');
5. A est une matrice de Van der Monde inversible si tous les points x_i sont différents, ce qui est le cas dans cette expérience. Et pourtant $A\alpha - b$ n'est pas égal 0. Ceci s'explique par le très mauvais conditionnement de la matrice A. Pour de plus grandes valeurs de n, par exemple $n = 20$, la matrice A devient singulière pour MATLAB , qui « prétend » que le rang de A est égal à 18 ; c'est là une mauvaise estimation du rang exact qui est égal à $n + 1$.

Le script de cet exercice est disponible sur le site web du livre sous le nom `ApproxScript1.m`.

Solution de l'exercice 3.2

Le script suivant est disponible sous le nom `ApproxScript2.m`.. Il fait appel à la fonction `condVanderMonde`.

```
%Conditionnement d'une matrice de Van der Monde
N=2:2:20;cd=[];
for n=N
    cd=[cd condVanderMonde(n)];
end;
plot(N,log(cd),'+-')
```

Listing 3.1 *condVanderMonde.m*

```
function y=condVanderMonde(n)
%calcule le conditionnement d'une matrice de Van der Monde.
%Les n+1 points sont uniformément répartis entre 0 et 1.
x=(0:n)'/n;
A=ones(length(x),1);
for k=1:length(x)-1
    A=[A x.^k];
end;
y=cond(A);
```

On note sur la figure 3.8 que ln(*cond*(*A*)) est une droite. On en déduit que *cond*(*A*) croit exponentiellement avec *n*.

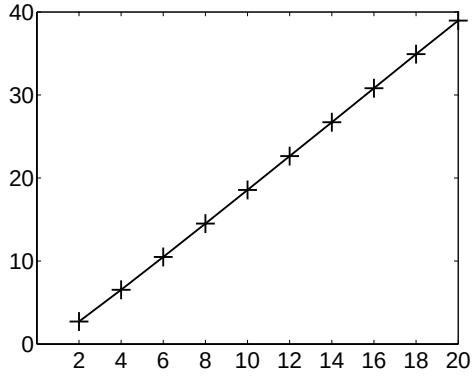


Figure 3.8 Logarithme du conditionnement d'une matrice de Van der Monde.

Comparer la fonction `condVanderMonde` avec la fonction suivante

Listing 3.2 *condVanderMondeBis.m*

```
function c=condVanderMondeBis(n)
%calcule le conditionnement d'une matrice de Van der Monde.
%Les n+1 points sont uniformément répartis entre 0 et 1.
x=(0:n)'/n;
A=ones(length(x),1);y=x;
for k=1:length(x)-1
    A=[A y];y=y.*x;
end;
c=cond(A);
```

Solution de l'exercice 3.3

```
%Base de Lagrange
n=10;x=(0:n)'/n;i=round(n/2);
deuxn=n;g=(0:deuxn)'/deuxn;
y=zeros(size(x));y(i)=1;cf=polyfit(x,y,n);
y0=polyval(cf,0)
```

Pour $n = 5$ ou 10 tout va bien, c'est-à-dire qu'on retrouve le fait que $\ell_{n/2}(x_0) = 0$. Mais pour $n = 20$, on trouve $\ell_{n/2}(x_0) = -1.0460$. Il s'agit là un encore d'un problème de conditionnement. MATLAB affiche d'ailleurs un message à ce sujet (pour $n = 20$). Voir le script `ApproxScript3.m`..

Solution de l'exercice 3.4

1. La fonction `dd.m` ci-dessous calcule des différences divisées d'une focntion, voir le fichier `dd.m`.

Listing 3.3 *dd.m*

```
function c=dd(x)
% x est formé des points xi
% c contient les différences divisées
c=test1(x);   %attention : "test1" est une fonction
              %définie dans un autre fichier ou "inline"
n=length(x);
for p=1:n-1
    for k=n:-1:p+1
        c(k)=(c(k)-c(k-1))/(x(k)-x(k-p));
    end;
end;
```

Pour passer le nom de la fonction à interpoler en argument d'appel de la fonction `dd`, il suffit de modifier les deux premières lignes :

```
function c=dd(x,f)
c=feval(f,x);
```

2.

Listing 3.4 *interpole.m*

```
function y=interpole(c,x,g)
%calcule l'interpolée de f sur la grille g
%connaissant les differences divisées c
%calculees aux points x
n=length(c);
y=c(n)*ones(size(g));
for k=n-1:-1:1
    y=c(k)+y.*(g-x(k));
end;
```

L'exécution du script ci-dessous produit la figure 3.9. Ce script est disponible sous le nom `ApproxScript4.m`.. De même que les fonctions `dd` et `interpole`.

```
n=20;x=(0:n)'/n;g=0:0.01:1;
c=dd(x);y=interpole(c,x,g);
yg=test1(g);plot(g,yg,g,y,'r+')
hold on;yx=test1(x);plot(x,yx,'O');hold off
```

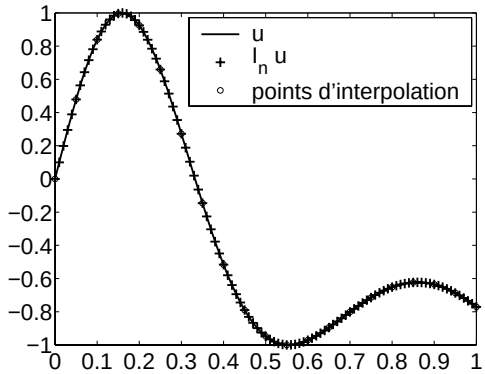


Figure 3.9 Calcul de l'interpolée de Lagrange aux points .

Solution de l'exercice 3.5

Le script de cet exercice est disponible sous le nom `ApproxScript5.m`. de même que la fonction `lebesgue`.

1. Calcul de la constante de Lebesgue.

Listing 3.5 *lebesgue.m*

```
function leb=lebesgue(x)
%calcule la constante de Lebesgue
%associee aux points du tableau x
n=length(x)-1;
xx=linspace(min(x),max(x),100);  %grille fine de 100 points
y=zeros(size(xx));
for i=1:n+1;
      %calcul de ℓ_i(x)
    l=zeros(size(x));l(i)=1;cf=polyfit(x,l,n);
    y=y+abs(polyval(cf,xx));
end;
leb=max(y);
```

2. Cas uniforme.

```
ind=[];lebE=[];
for n=10:5:30
    x=(-n/2:n/2)/n*2;      %points équidistants
    l=lebesgue(x)
    ind=[ind;n];lebE=[lebE;l];
end;
figure(1);plot(ind,log(lebE),'+-')
```

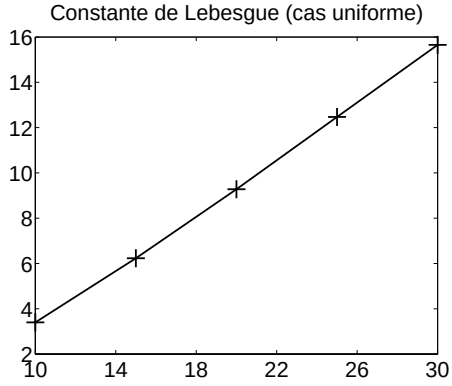


Figure 3.10 Constante de Lebesgue associée à des points équidistants : $n \mapsto \ln(\Lambda(n))$.

On note sur la figure 3.10 que $\ln(\Lambda(n))$ est une droite, de pente 1/ 2 environ. On en déduit que $\Lambda(n) \simeq e^{n/2}$.

3. Cas des points de Tchebycheff.

```
ind=[];lebT=[];
for n=10:5:30
    x=cos(pi*(.5+n:-1:0)/(n+1));      %points de Tchebycheff
    l=lebesgue(x);
    ind=[ind;n];lebT=[lebT;l];
end;
figure(2);plot(ind,exp(lebT),'+-')
```

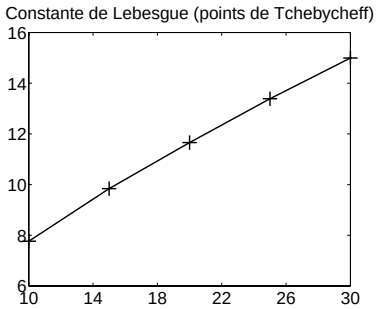


Figure 3.11 Constante de Lebesgue associée aux points de Tchebycheff : $n \mapsto e^{\Lambda(n)}$.

On note sur la figure 3.11 que $e^{\Lambda_T(n)}$ est une droite, de pente .6 environ. On en déduit que $\Lambda_T(n) \simeq .6 \ln(n)$. On peut prouver rigoureusement que $\Lambda_T(n) \simeq \frac{2}{\pi} \ln n$.

Solution de l'exercice 3.6

Pour la fonction f_1, les résultats avec $n = 30$ et $n = 40$ sont présentés sur la figure 3.12. Il semble que la méthode, si elle converge, converge lentement !

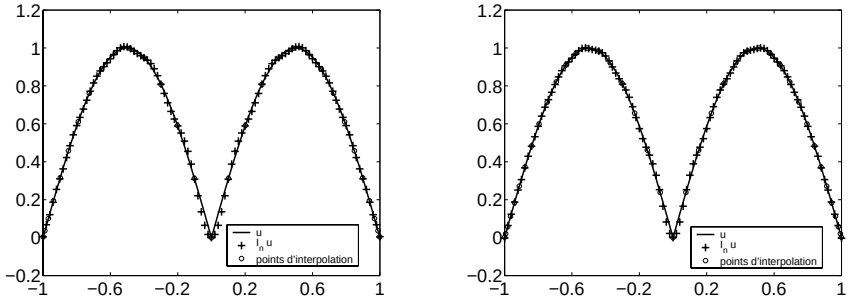


Figure 3.12 Interpolation aux points de Tchebycheff : $n = 30$ (à gauche) et $n = 40$ (à droite).

Pour la fonction f_2, les résultats avec $n = 5$ et $n = 10$ sont présentés sur la figure 3.13. Il semble que la méthode converge. On peut en fait prouver que l'interpolation aux points de Tchebycheff d'une fonction de classe $\mathcal{C}^1$ converge. Ce qui est le cas de f_2, mais pas de f_1.

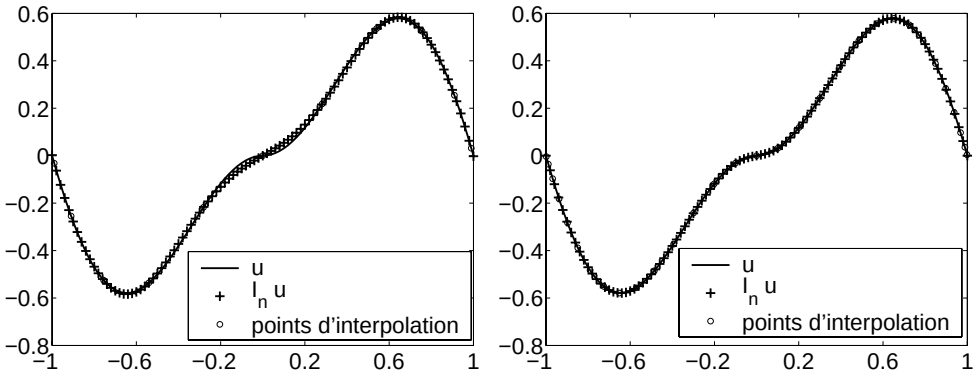


Figure 3.13 Interpolation aux points de Tchebycheff : $n = 10$ (à gauche) et $n = 30$ (à droite).

Solution de l'exercice 3.7

Les résultats avec $n = 10$, $n = 20$ et $n = 30$ sont présentés sur la figure 3.14. La méthode diverge car bien que la fonction soit très régulière, la constante de Lebesgue « explose » (voir la remarque 3.1).

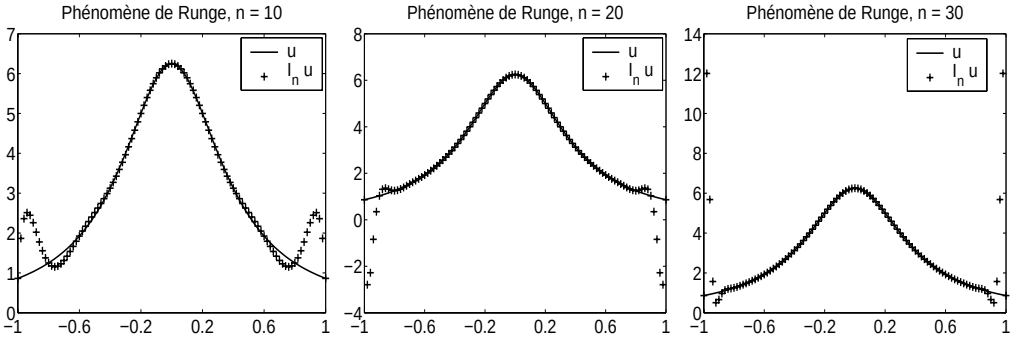


Figure 3.14 Le phénomène de Runge. De gauche à droite : interpolation en 11, 21 et 31 points uniformément répartis.

Solution de l'exercice 3.8

1. Calcul du polynôme d'interpolation au sens de Hermite par les différences divisées. On trouvera la fonction `ddHermite.m` sur le site web du livre. L'argument

d'entrée de cette fonction est un tableau `Tab` dont la première colonne est formée des points x_i et pour chaque i

– `Tab(i,1)` contient le point x_i.

– `Tab(i,2)` contient un entier α_i indiquant qu'on interpole la fonction ainsi que toutes ses dérivées jusqu'à l'ordre α_i.

– `Tab(i,3 :Tab(i,2)+3)`contient les valeurs de la fonction ainsi que ses derivees éventuelles.

La fonction `ddHermite.m` renvoie deux vecteurs

– Le premier vecteur contient les points x_i tenant compte de leur « multiplicité » : si l'on interpole au point x_i la fonction et ses dérivées à l'ordre α_i, ce point est copié $\alpha_i + 1$ fois dans `xx`.

– Le deuxième vecteur contient les différences divisées.

Avec ces deux vecteurs, on peut utiliser l'algorithme de Horner de la page 52 pour calculer $\mathcal{I}_n^H f(x)$, en tout point x.

2.

```
f=inline('cos(3*pi*x).*exp(-x)');
coll=[0 1/4 3/4 1]';
T=[coll zeros(size(coll)) f(coll)];
[xx,dd]=ddHermite(T);
%representer la fonction sur une grille fine
x=linspace(0,1,100);n=length(dd);
y=dd(n)*ones(size(x));
for k=n-1:-1:1
   y=dd(k)+y.*(x-xx(k));
end;
plot(x,y,x,f(x),'r');hold on;plot(coll,f(coll),'+')
```

Voir le script `ApproxScript8.m`. Sur la figure 3.15 (à gauche), les courbes se coupent seulement en quatre points ; elles sont très différentes en dehors de ces points.

3. Imposer que les dérivées doivent aussi coïncider, force le polynôme à « coller » plus à la fonction (figure 3.15, milieu). Il reste cependant une partie de l'intervalle $[0, 1]$ où les deux courbes ne sont pas proches.

4. En rajoutant un point d'interpolation pris dans une zone où l'approximation de f par le polynôme n'est pas satisfaisante, on obtient une approximation assez précise, voir la figure 3.15 (à doite).

Solution de l'exercice 3.9

Les résultats obtenus avec le script `ApproxScript9.m`. (voir le site web du livre) sont présentés sur la figure 3.16. Pour les valeurs de $m = 5(m' - 1)$ avec $m' \in \{1, 2, 3, 4, 5\}$, on calcule le polynôme p_m correspodant en faisant appel à la

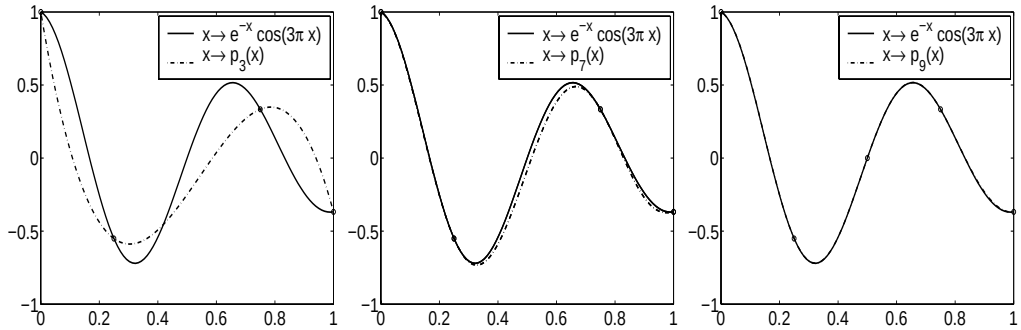


Figure 3.15 Le polynôme p_3 d'interpolation au sens de Lagrange aux points $0, 1/4, 3/4$ et 1, le polynôme p_7 d'interpolation au sens de Hermite aux mêmes points et le polynôme p_9 d'interpolation au sens de Hermite aux points $0, 1/4, 1/2, 3/4$ et 1. Les points d'interpolation sont représentés par des ronds.

fonction `ddHermite`. Les valeurs de p_m sur une grille uniforme de cent points de l'intervalle $[0, 1]$ sont stockées dans la colonne m' de la matrice `Y`.

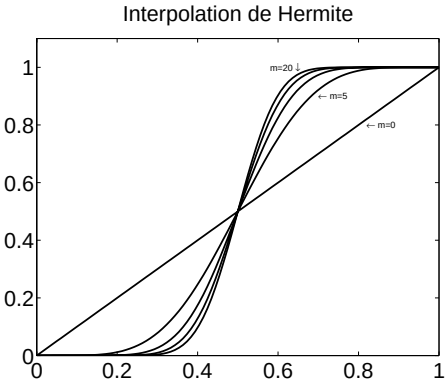


Figure 3.16 Polynômes d'interpolation au sens de Hermite.

Solution de l'exercice 3.10

Les équations obtenues forment un système linéaire de $n+1$ équations à $n+1$ inconnues (les coefficients du polynôme). Ce système a une solution unique si et seulement si l'unique solution du problème avec $f = 0$ est le polynôme nul. Ce qui est le cas puisque

– Le polynôme nul est clairement une solution.

– Si $p \in \mathbb{P}_n$ est une solution du problème. Soit $p(x_0) = 0$ et alors $p(x_i) = 0$ pour tout $i = 1, \ldots, n$ et p est le polynôme nul, puisqu'un polynôme de $\mathbb{P}_n$ ne peut pas avoir plus de n racines distinctes. Soit $p(x_0) \neq 0$ et alors p change de signe entre deux x_i consécutifs, donc s'annule en $n+1$ points distincts ; il est donc nul.

Notant $p(t) = \sum_{j=0}^{n} p_j t^j$, le système s'écrit

$$p(x_i) - (-1)^i p(x_0) = f(x_i) - (-1)^i f(x_0)\}, \qquad i = 1, \cdots, n+1$$

Le vecteur $a = (p_0, \cdots, p_n)^T$ est solution d'un système $Aa = b$ avec

$$A_{i,j} = x_i^j - (-1)^i x_0^j, \quad b_i = f(x_i) - (-1)^i f(x_0) \qquad (1 \leqslant i \leqslant n+1, 0 \leqslant j \leqslant n).$$

Voir la fonction `equiosc.m` sur le site web du livre. L'argument d'entrée de cette fonction est un vecteur contenant les valeurs x_i. La fonction calcule la matrice A et le vecteur b définis ci-desssus et renvoie les coefficients p_j du polynôme recherché.

Solution de l'exercice 3.11

La fonction `remez.m` calcule pour une valeur entière donnée n, la meilleure approximation polynomiale uniforme d'une fonction f. Trois choix des points d'initialisation de l'algorithme sont proposés : points de Tchebycheff, points uniformément répartis et points choisis au hasard. Le paramètre `tol` permet de relaxer l'égalité *(i)* de l'étape k de l'algorithme (c'est à dire qu'un test de la forme $a = b$ est remplacé par le test $|a - b| < \texttt{Tol}$).

Pour $n = 5$, $n = 10$ et $n = 15$ les meilleures approximations uniformes de la fonction définie sur $[0, 1]$ par $x \mapsto \sin(2\pi\cos(\pi x))$ sont présentées sur les figures 3.17.

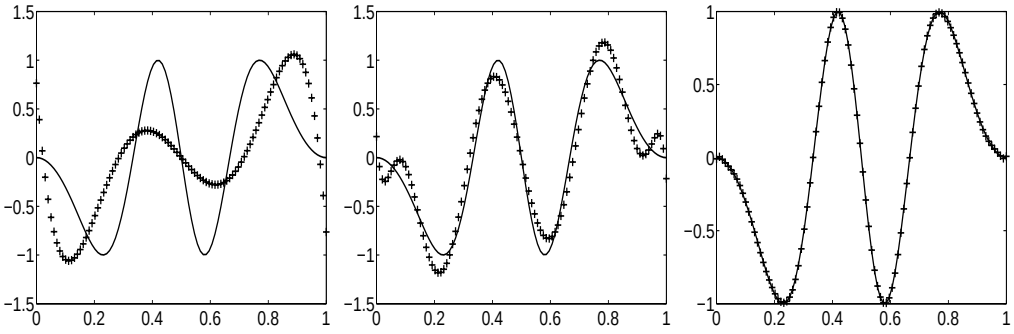


Figure 3.17 Meilleure approximation uniforme dans $\mathbb{P}_n$ de $f(x) = \sin(2\pi\cos(\pi x))$ représentée en trait continu. De gauche à droite : $n = 5$, $n = 10$ et $n = 15$.

Pour $n = 15$, l'algorithme initialisé par les points de Tchebycheff converge en 21 itérations. Initialisé par des points équidistants, l'algorithme converge en 28 itérations. Initialisé par des points choisis arbitrairement, l'algorithme ne converge (en général) pas au bout de 100 itérations. La fonction continue f a un développement en série de Tchebycheff, analogue au développement en série de Legendre (3.16)

$$f = \sum_{k=0}^{\infty} \hat{f}_k T_k.$$

On en déduit que

$$f - \sum_{k=0}^{n} \hat{f}_k T_k \simeq \hat{f}_{n+1} T_{n+1},$$

en négligeant le reste du développement. On note que T_{n+1} équioscille sur $[-1, 1]$ aux $n+2$ points $t_i = cos(i\frac{\pi}{n+1})$, $(i = 0, \cdots, n+1) : T_{n+1}(t_i) = \cos(i\pi) = (-1)^i$ et que les points x_i sont justement les images des points t_i par l'application affine qui envoie l'intervalle $[-1, 1]$ sur l'intervalle $[0, 1]$. On en déduit que $\sum_{k=0}^{n} \hat{f}_k T_k$ est proche de la meilleure approximation uniforme de f dans $\mathbb{P}_n$. C'est pourquoi ces points sont en général choisis pour initialiser l'algorithme de Remez.

Solution de l'exercice 3.12

Le script `mdc.m` calcule la meilleure approximation au sens des moindres carrés, sur $[0, 1]$, d'une fonction donnée. L'instruction `p=polyfit(x,y,n)` renvoie un tableau `p` contenant les coefficients du polynôme de degré inférieur ou égal à `n` interpolant les valeurs `y(i)` aux points `x(i)`. Pour calculer les valeurs prises par ce polynôme sur une grille de points, on utilise la commande `polyval`. L'exécution du script `mdc.m` fournit la valeur $n = 10$. Le polynôme de meilleure approximation est représenté sur la figure 3.18, des '+' désignent le nuage de points (x_i, y_i).

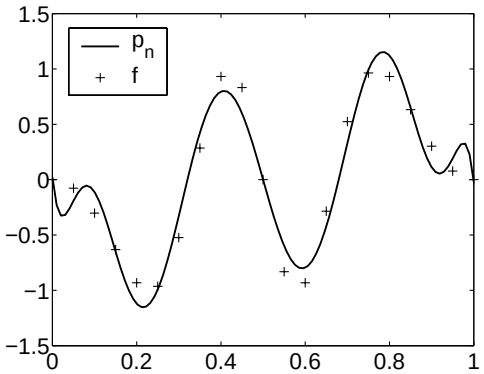


Figure 3.18 Approximation au sens des moindres carrés dans $\mathbb{P}_{10}$. La fonction à approcher est représentée en trait continu.

Solution de l'exercice 3.13

On présente uniquement le calcul de la spline constante par morceaux.

Listing 3.6 *spline0.m*

```
n0=10;E=[];N=[];
for i=1:10,
    n=i*n0;E=[E;erreurS0(n)];N=[N;n];
end;
```

```
loglog(N,E,'-+');xlabel('log n');ylabel('log Erreur');
fprintf('pente de la droite = %g ',log(E(end)/(E(1)))/log(N(end)/N(1)));
function y=erreurS0(n)
x=(0:n)'/n;h=1/n;fx=f(x);
%sur chaque intervalle [x_i, x_{i+1}], on calcule p_i
y=[];
for i=1:n
   Ii=linspace(x(i),x(i+1),20);
   fi=f(Ii);
   Si=f(.5*(x(i)+x(i+1)));
   y=[y norm(Si-fi,'inf')];
end
y=max(y);


function y=f(x)
y=sin(4*pi*x);
```

L'exécution de ce script produit la figure 3.19. La pente de la droite, calculée par MATLAB est d'environ -0.971325, ce qui est une approximation acceptable de la valeur exacte -1 donné par (3.26).

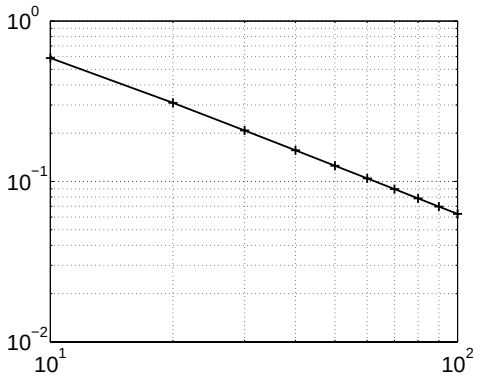


Figure 3.19 Courbe $\ln n \mapsto \ln \|f - S_0\|_\infty$.

Solution de l'exercice 3.14

Le script `spline1.m` (voir le site web du livre) produit la figure 3.20. La pente de la droite, calculée par MATLAB est d'environ -1.965, ce qui est une approximation acceptable de la valeur exacte -2 donné par (3.27).

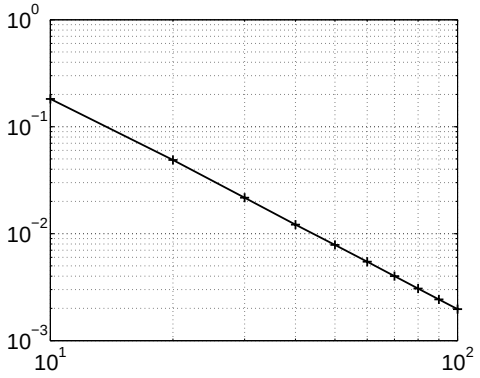


Figure 3.20 Courbe $\ln n \mapsto \ln \|f - S_1\|_\infty$.

Solution de l'exercice 3.15

Voir le script `spline3.m`.

Les résultats produits pour $n = 5$ et $n = 10$ sont présentés sur la figure 3.21.

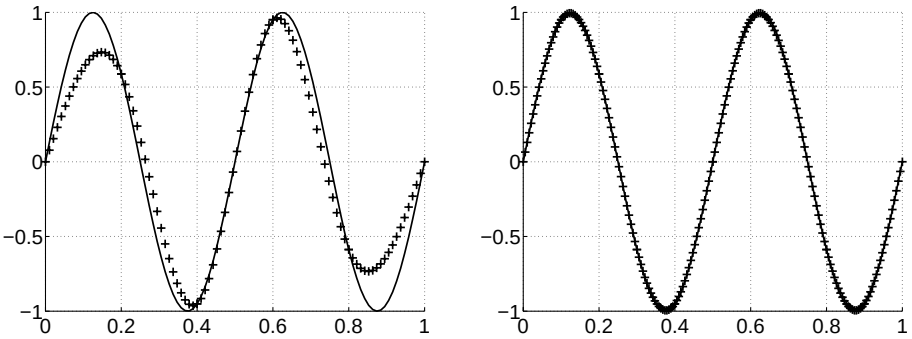


Figure 3.21 Splines cubiques : $n = 5$ (à gauche) et $n = 10$ (à droite).

BIBLIOGRAPHIE

[Allaire et Kaber, 2002 (1)] G. ALLAIRE, S.M. KABER : *Algèbre linéaire numérique. Cours et exercices*, Ellipses, Paris, 2002.

[Allaire et Kaber, 2002 (2)] G. ALLAIRE, S.M. KABER : *Introduction à Scilab. Exercices pratiques corrigés d'algèbre linéaire*, Ellipses, Paris, 2002.

[Bernardi, Maday et Rapetti, 2004] C. BERNARDI, Y. MADAY, F. RAPETTI : *Discrétisations variationnelles de problèmes aux limites elliptiques, collection S.M.A.I. Mathématiques et Applications*, vol. 45, Springer Verlag, Paris, 2004.

[Cohen, 2003] A. COHEN : *Numerical Analysis of Wavelet Methods, Studies in Mathematics and its Applications, 32, North-Holland, Amsterdam, 2003.*

[Ciarlet, 1982] P. G. CIARLET : *Introduction à l'analyse numérique matricielle et à l'optimisation*, Masson, Paris, 1982.

[Crouzeix et Mignot, 1989] M. CROUZEIX, A. MIGNOT : *Analyse numérique des équations différentielles*, Masson, Paris, 1989.

[Delabrière et Postel, 2004] S. DELABRIÈRE, M. POSTEL : *Méthodes d'approximation. Équations différentielles. Applications Scilab*, Ellipses, Paris, 2004.

[Dumas, 1999] L. DUMAS : *Modélisation à l'oral de l'agrégation*, Ellipses, Paris, 1999.

[Joly, 2004] P. JOLY : *Analyse numérique matricielle*, Cassini, Paris, 2004.

[Théodor et Lascaux, 1985] R. THÉODOR, P. LASCAUX : *Analyse numérique matricielle appliquée à l'art de l'ingénieur*, Masson, Paris, 1985.

[Schwartz, 1980] L. SCHWARTZ : *Analyse, topologie générale et analyse fonctionnelle*, Hermann, Paris, 1980.

Projet **4**

Étude d'un modèle de convection-diffusion par éléments finis

Fiche du projet

Difficulté :	1
Notions développées :	Équation de convection-diffusion, méthode des éléments finis, stabilisation d'un schéma numérique
Domaines d'application :	Problèmes de couche limite

On cherche à calculer une approximation de la solution $u : [0,1] \longrightarrow \mathbb{R}$ du problème suivant

$$\begin{cases} -\varepsilon u''(x) + \lambda u'(x) = f(x), \\ u(0) = 0, \\ u(1) = 0. \end{cases} \tag{4.1}$$

La fonction f et les réels $\varepsilon > 0$ et λ sont donnés de sorte qu'il existe une unique solution à ce problème qu'on pourra approcher par une fonction continue, polynomiale par morceaux.

4.1 MODÉLISATION

L'équation différentielle du problème (4.1) est une équation de convection-diffusion. Elle peut modéliser la concentration d'une espèce chimique transportée dans un fluide de vitesse λ, le paramètre ε modélisant la diffusivité de l'espèce chimique.

Le rapport $\theta = \lambda/\varepsilon$ mesure l'importance des phénomènes de convection par rapport aux phénomènes de diffusion. Pour les grandes valeurs de ce rapport, la résolution numérique du problème (4.1) est délicate. La création et la disparition de l'espèce chimique sont prises en compte dans la fonction f, qui en toute généralité peut dépendre de l'inconnue u. Cependant, par souci de simplification, on ne fera dépendre f que de la position x et on prendra λ et ε constants.

4.2 FORMULATION VARIATIONNELLE DU PROBLÈME

On admettra, qu'une solution « assez régulière » u du problème aux limites (4.1) est aussi solution du problème suivant

$$\begin{cases} \text{Trouver } u \in H_0^1(0,1) \text{ tel que, pour tout } v \in H_0^1(0,1), \text{ on ait} \\ \varepsilon \displaystyle\int_0^1 u'(x)v'(x)dx + \lambda \int_0^1 u'(x)v(x)dx = \int_0^1 f(x)v(x)dx. \end{cases} \tag{4.2}$$

On a noté ici $H_0^1(0,1)$ l'ensemble des fonctions $v : [0,1] \to \mathbb{R}$ telles que les intégrales $\displaystyle\int_0^1 |v|^2$ et $\displaystyle\int_0^1 |v'|^2$ soient finies et $v(0) = v(1) = 0$. Et réciproquement : toute solution régulière de (4.2) est une solution de (4.1). Le problème (4.2) est appelé « formulation variationnelle » du problème (4.1). La méthode des éléments finis est basée sur le calcul de la solution de la formulation variationnelle (4.2) et non pas sur la discrétisation « directe » de l'équation (4.1) comme c'est le cas dans la méthode des différences finies.

Étant donné un entier non nul n, on divise l'intervalle $[0,1]$ en $n+1$ sous-intervalles I_i. Étant donné un entier non nul ℓ, on note $\mathbb{P}_\ell(I_i)$ l'ensemble des polynômes de degré inférieur ou égal à ℓ sur I_i et $\mathcal{V}_\ell^h$ l'ensemble de fonctions continues sur $[0,1]$ et dont la restriction à chaque intervalle I_i est un polynôme de degré inférieur ou égal à ℓ. On présente sur la figure 4.1 un exemple de fonctions de $\mathcal{V}_1^h$ et $\mathcal{V}_2^h$.

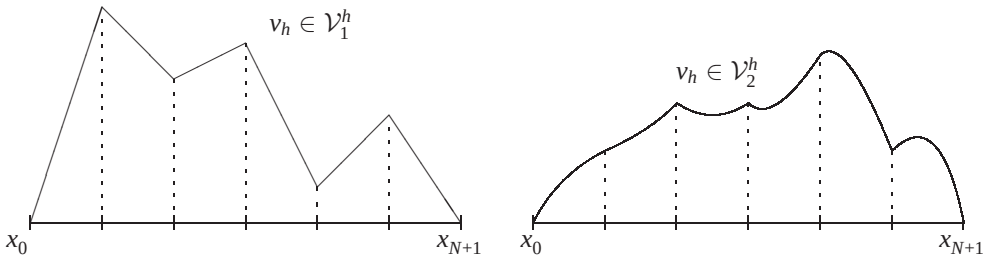


Figure 4.1 Exemples de fonction de $\mathcal{V}_1^h$ (à gauche) et de $\mathcal{V}_2^h$ (à droite).

Les méthodes d'éléments finis consistent à chercher une approximation $u_h \in \mathcal{V}_\ell^h$ de la fonction u, solution du problème suivant (comparer avec (4.2))

$$\begin{cases} \text{Trouver } u_h \in \mathcal{V}_\ell^h \text{ tel que, pour tout } v_h \in \mathcal{V}_\ell^h, \text{ on ait} \\ \varepsilon \displaystyle\int_0^1 u_h'(x)v_h'(x)dx + \lambda \int_0^1 u_h'(x)v_h(x)dx = \int_0^1 f(x)v_h(x)dx. \end{cases} \tag{4.3}$$

Les intégrales apparaissant à gauche dans (4.3) se calculent facilement puisqu'elles font intervenir des produits de polynômes. Ce n'est pas le cas de l'intégrale $\int_0^1 f(x)v_h(x)dx$ dont le calcul explicite n'est pas toujours possible. On procède dans ce cas à une intégration numérique. Nous utiliserons deux formules d'intégration numérique.

– *La formule des trapèzes.*

$$\int_\alpha^\beta g(x)dx \simeq (\beta - \alpha)\frac{g(\alpha) + g(\beta)}{2}.$$

Cette méthode est d'ordre 1, c'est-à-dire que si $g \in \mathbb{P}_1([\alpha, \beta])$ la formule est exacte.

– *La formule de Simpson.*

$$\int_\alpha^\beta g(x)dx \simeq \frac{\beta - \alpha}{6}\left(g(\alpha) + 2g(\frac{\alpha + \beta}{2}) + g(\beta)\right)$$

Cette méthode est d'ordre 3, c'est-à-dire que si $g \in \mathbb{P}_3([\alpha, \beta])$ la formule est exacte.

Dans ce chapitre, nous allons comparer deux méthodes d'éléments finis pour résoudre le problème de convection-diffusion. La première méthode est dite $P1$ car elle utilise des fonctions de $\mathcal{V}_1^h$ et la deuxième est une méthode $P2$ puisque l'espace d'approximation est $\mathcal{V}_2^h$. Pour valider les calculs numériques, on aura besoin de connaître la solution exacte du problème, ce qui se fait aisément dans le cas où la fonction f est constante, c'est le but de l'exercice suivant.

Exercice 4.1 Calcul de la solution exacte

On suppose la fonction f constante, non nulle.

1. Calculer la solution exacte u du problème (4.1).
2. Montrer qu'il existe un point $x_\theta \in]0, 1[$ ne dépendant que du rapport $\theta = \lambda/\varepsilon$ tel que la fonction $\dfrac{\lambda}{f}u$ soit strictement croissante (resp. décroissante) sur $]0, x_\theta[$ (resp. $]x_\theta, 1[$). Calculer $\displaystyle\lim_{|\theta|\to+\infty} x_\theta$.
3. Pour $\lambda > 0$ fixé, on étudie le comportement de la solution u quand ε tend vers 0^+ (et donc $\theta \to +\infty$). Calculer $u(x_\theta)$, puis $\displaystyle\lim_{\varepsilon\to0^+} u(x_\theta)$. Montrer que

$$\lim_{\varepsilon\to0^+}\lim_{x\to1} u(x) \neq \lim_{x\to1}\lim_{\varepsilon\to0^+} u(x).$$

Expliquer ce que signifie l'affirmation « pour ε petit, la solution du problème différentiel (4.1) a une couche limite au voisinage du point $x = 1$ ».

4. Écrire une fonction permettant de calculer la solution u sur une grille (un tableau) de points donnée. Tracer u pour $f = 1$, $\lambda \in \{-1, 1\}$ et $\varepsilon \in \{1, 1/2, 10^{-1}, 10^{-2}\}$. Commenter les résultats.

La solution de cet exercice se trouve page 98.

4.3 UNE MÉTHODE D'ÉLÉMENTS FINIS *P*1

Pour $n \in \mathbb{N}^*$, on pose $h = 1/(n+1)$ et définit les points $x_k^{(1)} = kh$ $(k = 0, \ldots, n+1)$ et les intervalles $I_k =]x_k^{(1)}, x_{k+1}^{(1)}[$, $(k = 0, \ldots, n)$. Soit a la forme bilinéaire définie par

$$a(u, v) = \varepsilon \int_0^1 u'(x)v'(x)dx + \lambda \int_0^1 u'(x)v(x)dx, \tag{4.4}$$

on cherche une approximation $u_h^{(1)} \in \mathcal{V}_1^h$ de la fonction u, solution du problème (4.3) avec $\ell = 1$:

$$\begin{cases} \text{Trouver } u_h^{(1)} \in \mathcal{V}_1^h \text{ tel que, pour tout } v_h \in \mathcal{V}_1^h, \text{ on ait} \\ a(u_h^{(1)}, v_h) = \displaystyle\int_0^1 f(x)v_h(x)dx. \end{cases} \tag{4.5}$$

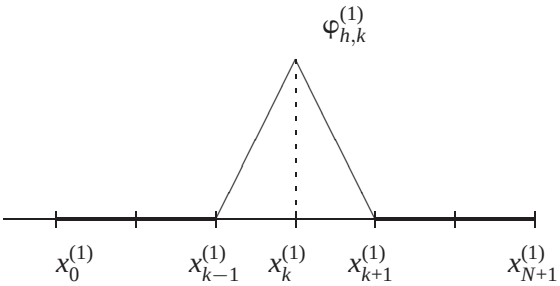


Figure 4.2 $\varphi_{h,k}^{(1)}$ une fonction de base de $\mathcal{V}_1^h$.

On définit les k fonctions "chapeaux" $\varphi_{h,k}^{(1)}$ $(k = 1, \ldots, n)$ par (voir la figure 4.2)

$$\varphi_{h,k}^{(1)} \in \mathcal{V}_1^h \text{ et } \varphi_{h,k}^{(1)}(x_j) = \delta_{j,k}, \quad \forall j = 1, \ldots, n.$$

Noter que le support de la fonction $\varphi_{h,k}^{(1)}$ est égal à la réunion des deux intervalles I_{k-1} et I_k.

Exercice 4.2

1. Montrer que les $\varphi_{h,k}^{(1)}$ forment une base de $\mathcal{V}_1^h$.
2. En déduire que le problème (4.5) est équivalent au problème

$$\begin{cases} \text{Trouver } u_h^{(1)} \in \mathcal{V}_1^h \text{ tel que, pour tout } (k = 1, \ldots, n), \text{ on ait} \\ a(u_h^{(1)}, \varphi_{h,k}^{(1)}) = \displaystyle\int_0^1 f(x)\varphi_{h,k}^{(1)}(x)dx. \end{cases} \tag{4.6}$$

3. En écrivant $u_h^{(1)}$ dans la base des $\varphi_{h,k}^{(1)}$; $u_h^{(1)} = \sum_{m=1}^n \alpha_m \varphi_{h,m}^{(1)}$, montrer que $\alpha_k = u_h(x_k^{(1)})$ puis que le vecteur $\tilde{u}_h^{(1)} = (u_h^{(1)}(x_1^{(1)}), \ldots, u_h^{(1)}(x_n^{(1)}))^T$ est solution d'un système linéaire

$$A_h^{(1)}\tilde{u}_h^{(1)} = b_h^{(1)} \tag{4.7}$$

où $A_h^{(1)}$ est la matrice réelle de taille $n \times n$ définie par

$$(A_h^{(1)})_{k,m} = a(\varphi_{h,m}^{(1)}, \varphi_{h,k}^{(1)}), \quad (1 \leqslant m, k \leqslant n)$$

et $b_h^{(1)}$ le vecteur de $\mathbb{R}^n$ défini par

$$(b_h^{(1)})_k = \int_0^1 f(x)\varphi_{h,k}^{(1)}(x)dx, \quad (1 \leqslant k \leqslant n).$$

En déduire que $A_h^{(1)} = \varepsilon B_h^{(1)} + \lambda C_h^{(1)}$, où $B_h^{(1)}$ est une matrice symétrique tridiagonale et $C_h^{(1)}$ une matrice antisymétrique tridiagonale.

4. Montrer que la matrice symétrique $B_h^{(1)}$ est définie positive, c'est-à-dire que pour tout $x \in \mathbb{R}^n$, on a $\langle B_h^{(1)}x, x\rangle \geqslant 0$, avec égalité si et seulement si le vecteur x est nul. Cette propriété est très utile dans l'analyse numérique des problèmes matriciels. Elle implique en particulier que la matrice est inversible.
5. Montrer que $\langle A_h^{(1)}x, x\rangle = \langle B_h^{(1)}x, x\rangle$. En déduire que la matrice $A_h^{(1)}$ est inversible.

La solution de cet exercice se trouve page 99.

Le système (4.6) admet donc une solution unique que nous allons calculer en résolvant le système linéaire (4.7).

Exercice 4.3 Calcul numérique de la solution $P1$

1. Montrer que

$$B_h^{(1)}=\frac{1}{h}\begin{pmatrix} 2 & -1 & 0 & \dots & \dots & 0 \\ -1 & 2 & -1 & 0 & & \vdots \\ 0 & \ddots & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \vdots \\ \vdots & & 0 & -1 & 2 & -1 \\ 0 & \dots & \dots & 0 & -1 & 2 \end{pmatrix}, \quad C_h^{(1)}=\frac{1}{2}\begin{pmatrix} 0 & 1 & 0 & \dots & \dots & 0 \\ -1 & 0 & 1 & 0 & & \vdots \\ 0 & \ddots & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \vdots \\ \vdots & & 0 & -1 & 0 & 1 \\ 0 & \dots & \dots & 0 & -1 & 0 \end{pmatrix}.$$

2. Écrire une fonction calculant la matrice $A_h^{(1)}$ (arguments d'entrée : n, ε et λ).
3. Écrire une fonction calculant le second membre $b_h^{(1)}$ (arguments d'entrée : n et f). On utilisera la formule des trapèzes pour calculer les composantes du vecteur $b_h^{(1)}$.
4. Pour $\varepsilon = 0.1$, $\lambda = 1$, $f = 1$ et $n \in \{10, 20\}$, calculer la solution $\tilde{u}_h$ de (4.7) et comparer la avec la solution exacte u.
5. Étude de l'erreur. On fixe toujours $\varepsilon = 0.1$, $\lambda = 1$ et $f = 1$. Pour n variant de 10 à 100 (par pas de 10), tracer les courbes $\log n \mapsto \log \|e_n^{(1)}\|_\infty$, où $e_n^{(1)} \in \mathbb{R}^n$ est le vecteur erreur défini par $(e_n^{(1)})_k = \tilde{u}_h^{(1)}(k) - u(x_k^{(1)})$. En déduire une loi de décroissance de $\|e_n^{(1)}\|_\infty$ de la forme $\|e_n^{(1)}\|_\infty \simeq \mathit{Constante}/n^{s_1}$ où $s_1 > 0$ est à déterminer.

La solution de cet exercice se trouve page 101.

La méthode des éléments finis $P1$ semble donc bien indiquée pour résoudre le problème de convection-diffusion. Les choses ne sont malheureusement pas aussi simples. Pour $\lambda = 1$, $f = 1$, $\varepsilon = 0.01$ et $n = 10$, on obtient la figure 4.3 qui montre que $u_h^{(1)}$ est une fonction oscillante qui n'est clairement pas une bonne approximation de u, en particulier dans la couche limite.

On verra à la section suivante si une approximation par éléments finis d'ordre élevé supprime ces oscillations. Pour le moment, on revient à la méthode $P1$ pour répondre à la question suivante : à ε donné, déterminer le nombre d'intervalles nécessaires pour bien représenter la couche limite.

Exercice 4.4

On fixe $\lambda = 1$ et $f = 1$. Pour diverses "petites" valeurs de ε par exemple dans la plage $[0,005; 0,02]$, déterminer le nombre $n \equiv n(\varepsilon)$ à partir duquel la solution numérique semble être une approximation raisonnable de la solution exacte dans la couche limite. On considérera que c'est le cas si la solution numérique n'oscille pas et est proche de la solution exacte. On procédera ainsi : à ε fixé,

on lance le programme pour $n = 10$, puis $20, 30$, etc. Pour chaque valeur de n, on représente sur un graphique la solution exacte et la solution approchée. Au vu de ce graphique, on décide de la qualité de l'approximation. Pour chaque valeur de n, on calculera la valeur $P = \frac{|\lambda|h}{2\varepsilon}$ appelée nombre de Peclet de la grille. Commenter.

La solution de cet exercice se trouve page 103.

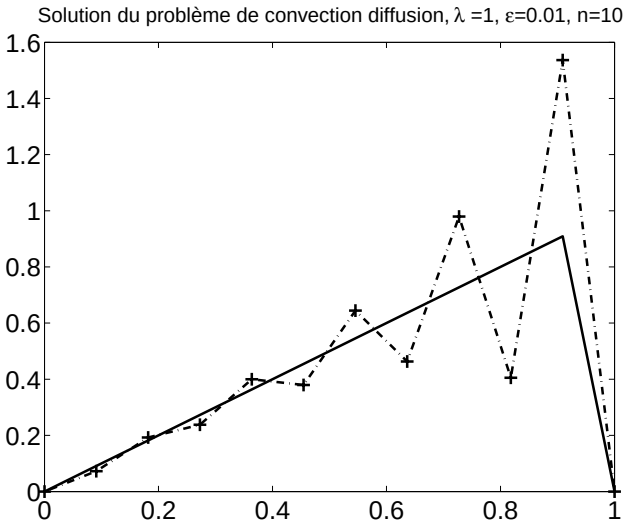


Figure 4.3 Approximation de la solution du problème de convection-diffusion par une méthode d'éléments finis $P1$, $\varepsilon = 0.01$, $\lambda = 1$ et $n = 10$.

4.4 UNE MÉTHODE D'ÉLÉMENTS FINIS *P*2

Pour une meilleure approximation, nous allons associer à chaque sommet du maillage non plus une fonction affine par morceaux, mais une fonction quadratique par morceaux.

Pour $n \in \mathbb{N}^*$, on pose $h = 1/(n+1)$ et définit les points $x_k^{(2)} = kh/2$ $(k = 0, \ldots, 2(n+1))$. Noter que $x_{2k}^{(2)} = x_k^{(1)}$ et que les intervalles $I_k =]x_{2k}^{(2)}, x_{2k+2}^{(2)}[$ $(k = 0, \ldots, n)$ sont ceux de la section précédente. Autrement dit, on garde le même nombre d'intervalles, mais on rajoute à chaque intervalle un nouveau "nœud", à savoir le centre de l'élément.

On cherche cette fois une approximation $u_h \in \mathcal{V}_2^h$ de la fonction u, solution du problème (4.3) avec $\ell = 2$

$$\begin{cases} \text{Trouver } u_h^{(2)} \in \mathcal{V}_2^h \text{ tel que, pour tout } v_h \in \mathcal{V}_2^h, \text{ on ait} \\ a(u_h^{(2)}, v_h) = \displaystyle\int_0^1 f(x)v_h(x)dx. \end{cases} \tag{4.8}$$

Comme dans la section précédente on commence par déterminer une base simple de $\mathcal{V}_2^h$. Sur chaque intervalle I_k, on définit les trois polynômes quadratiques de Lagrange associés aux points $x_{2k}^{(2)}$, $x_{2k+1}^{(2)}$ et $x_{2k+2}^{(2)}$.

$$\begin{cases} \psi_{h,k}^{(-)}(x) = \quad 2(x - x_{2k+1}^{(2)})(x - x_{2k+2}^{(2)})/h^2, \\ \psi_{h,k}^{(0)}(x) = -4(x - x_{2k}^{(2)})(x - x_{2k+2}^{(2)})/h^2, \\ \psi_{h,k}^{(+)}(x) = \quad 2(x - x_{2k}^{(2)})(x - x_{2k+1}^{(2)})/h^2. \end{cases}$$

À chaque nœud $x_k^{(2)}$ du maillage, on associe la fonction $\varphi_{h,k}^{(2)} \in \mathcal{V}_2^h$ définie par (voir aussi la figure 4.4)

$$\varphi_{h,2k+1}^{(2)}(x) = \begin{cases} \psi_{h,k}^{(0)}(x) & \text{pour } x \in I_k, \\ 0 & \text{sinon.} \end{cases} \quad \text{et} \quad \varphi_{h,2k}^{(2)}(x) = \begin{cases} \psi_{h,k}^{(-)}(x) & \text{pour } x \in I_k, \\ \psi_{h,k-1}^{(+)}(x) & \text{pour } x \in I_{k-1}, \\ 0 & \text{sinon.} \end{cases}$$

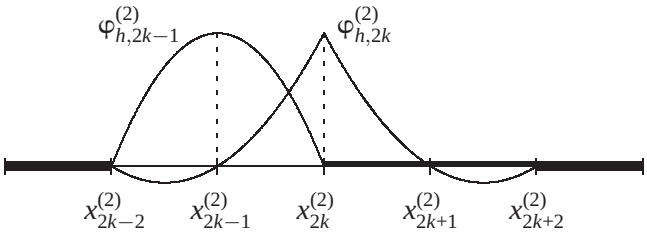


Figure 4.4 Les fonctions de base de $\mathcal{V}_2^h$.

Noter que le support de la fonction $\varphi_{h,k}^{(2)}$ est égal à I_k ou à la réunion des deux intervalles I_{k-1} et I_k, suivant la parité de k. Comme les $\varphi_{h,k}^{(2)}$ forment une base de $\mathcal{V}_2^h$, le problème (4.8) est équivalent au problème

$$\begin{cases} \text{Trouver } u_h^{(2)} \in \mathcal{V}_2^h \text{ tel que, pour tout } (k = 1, \ldots, 2n+1), \text{ on ait} \\ a(u_h^{(2)}, \varphi_{h,k}^{(2)}) = \displaystyle\int_0^1 f(x)\varphi_{h,k}^{(2)}(x)dx. \end{cases} \tag{4.9}$$

On écrit $u_h^{(2)}$ dans la base des $\varphi_{h,k}^{(2)}$: $u_h^{(2)} = \sum_{m=1}^{2n+1} \alpha_m \varphi_{h,m}$. Comme dans le cas $P1$, on montre que $\alpha_m = u_h^{(2)}(x_m^{(2)})$ et que le vecteur $\tilde{u}_h^{(2)} = (\alpha_1, \ldots, \alpha_{2n+1})^T$ est solution d'un système linéaire

$$A_h^{(2)} \tilde{u}_h^{(2)} = b_h^{(2)} \tag{4.10}$$

où $A_h^{(2)}$ est la matrice de taille $(2n+1) \times (2n+1)$ définie par

$$(A_h^{(2)})_{k,m} = a(\varphi_{h,m}^{(2)}, \varphi_{h,k}^{(2)}), \quad (1 \leqslant m, k \leqslant 2n+1)$$

et b_h le vecteur de $\mathbb{R}^{2n+1}$ défini par

$$(b_h^{(2)})_k = \int_0^1 f(x)\varphi_{h,k}^{(2)}(x)dx, \quad (1 \leqslant k \leqslant 2n+1).$$

Exercice 4.5

On écrit $A_h^{(2)} = \varepsilon B_h^{(2)} + \lambda C_h^{(2)}$. La matrice $B_h^{(2)}$ est-elle symétrique, tridiagonale ? La matrice $C_h^{(2)}$ est-elle antisymétrique, tridiagonale ? Montrer que la matrice $A_h^{(2)}$ est inversible.

La solution de cet exercice se trouve page 103.

Le système (4.9) admet donc une solution unique qu'on peut calculer en résolvant le système linéaire (4.10).

Exercice 4.6 Calcul numérique de la solution *P*2

1. Montrer que $B_h^{(2)}$ et $C_h^{(2)}$ ont la forme suivante, donnée ici pour $n = 3$

$$B_h^{(2)} = \frac{1}{3h}\begin{pmatrix} 16 & -8 & 0 & 0 & 0 & 0 & 0 \\ -8 & 14 & -8 & 1 & 0 & 0 & 0 \\ 0 & -8 & 16 & -8 & 0 & 0 & 0 \\ 0 & 1 & -8 & 14 & -8 & 1 & 0 \\ 0 & 0 & 0 & -8 & 16 & -8 & 0 \\ 0 & 0 & 0 & 1 & -8 & 14 & -8 \\ 0 & 0 & 0 & 0 & 0 & -8 & 16 \end{pmatrix},$$

$$C_h^{(2)} = \frac{1}{3}\begin{pmatrix} 0 & -2 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & -2 & 1/2 & 0 & 0 & 0 \\ 0 & 2 & 0 & -2 & 0 & 0 & 0 \\ 0 & -1/2 & 2 & 0 & -2 & 1/2 & 0 \\ 0 & 0 & 0 & 2 & 0 & -2 & 0 \\ 0 & 0 & 0 & -1/2 & 2 & 0 & -2 \\ 0 & 0 & 0 & 0 & 0 & 2 & 0 \end{pmatrix}.$$

2. Écrire une fonction calculant la matrice $A_h^{(2)}$ (arguments d'entrée : n, ε et λ).
3. Écrire une fonction calculant le second membre $b_h^{(2)}$ (arguments d'entrée : n et f). On utilisera la formule de Simpson pour calculer les composantes du vecteur $b_h^{(2)}$.
4. Étude de l'erreur. On fixe toujours $\varepsilon = 0.1$, $\lambda = 1$ et $f = 1$. Pour n variant de 10 à 100 (par pas de 10), tracer les courbes $\log n \mapsto \log \|e_n^{(2)}\|_\infty$, où $e_n^{(2)} \in \mathbb{R}^{2n+1}$ est le vecteur erreur défini par $(e_n^{(2)})_k = \tilde{u}_h^{(2)}(k) - u(x_k^{(2)})$. En déduire une loi de décroissance de $\|e_n^{(2)}\|_\infty$ de la forme $e_n^{(2)} \simeq Constante/n^{s_2}$ où $s_2 > 0$ est à déterminer.

La solution de cet exercice se trouve page 103.

4.5 UNE TECHNIQUE DE STABILISATION

On appelle stabilisation un moyen de supprimer les oscillations observées sur la figure 4.3. En lançant le programme *P*2 avec le même jeu de paramètres ($\lambda = 1$, $f = 1$, $\varepsilon = 0.01$ et $n = 10$) on trouve la figure 4.5. À première vue, les oscillations persistent. Cependant, si on ne s'intéresse qu'aux valeurs de la solution aux extrémités des intervalles (les points $x_{2k}^{(2)}$), la courbe obtenue est une approximation non oscillante de la solution exacte. On se propose de vérifier ce phénomène et de l'expliquer. Commençons par définir un moyen de calculer les valeurs de $u_h^{(2)}$ aux bords des intervalles, sans calculer les valeurs aux milieux des intervalles.

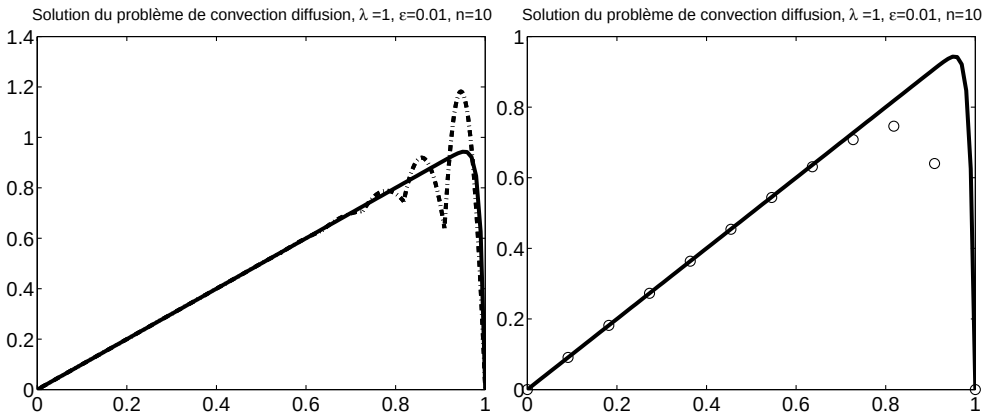


Figure 4.5 Approximation de la solution du problème de convection-diffusion pour $\varepsilon = 0.01$, $\lambda = 1$ et $n = 10$. Solution par la méthode d'éléments finis *P*2 (à gauche), même solution représentée uniquement aux extrémités des intervalles (à droite).

4.5.1 Calcul de la solution aux extrémités des intervalles

Soit $\widehat{A}_h^{(2)}$ la matrice obtenue à partir de la matrice $A_h^{(2)}$ en permutant les lignes, pour mettre aux premières places les lignes d'indices pairs, puis en faisant la même opération pour les colonnes. De même, on permute les composantes du vecteur $b_h^{(2)}$ en mettant ses composantes paires aux premières places ; on obtient ainsi un vecteur $\widehat{b}_h^{(2)}$. On définit aussi un vecteur $\widehat{u}_h^{(2)}$ à partir du vecteur $\tilde{u}_h^{(2)}$. Le système (4.10) est équivalent au système $\widehat{A}_h^{(2)}\widehat{u}_h^{(2)} = \widehat{b}_h^{(2)}$ qu'on aurait obtenu directement à partir de la formulation variationnelle en changeant la numérotation des inconnues. On peut décomposer $\widehat{A}_h^{(2)}$, $\widehat{b}_h^{(2)}$ et $\widehat{u}_h^{(2)}$ en

$$\widehat{A}_h^{(2)} = \begin{bmatrix} A & B \\ C & D \end{bmatrix}, \quad \widehat{b}_h^{(2)} = \begin{bmatrix} c \\ d \end{bmatrix}, \quad \widehat{u}_h^{(2)} = \begin{bmatrix} v \\ w \end{bmatrix},$$

avec $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times(n+1)}$, $C \in \mathbb{R}^{(n+1)\times n}$, $D \in \mathbb{R}^{(n+1)\times(n+1)}$, $c \in \mathbb{R}^n$, $d \in \mathbb{R}^{n+1}$, $v \in \mathbb{R}^n$, $w \in \mathbb{R}^{n+1}$ et

$$A_{i,j} = a(\varphi^{(2)}_{h,2j}, \varphi^{(2)}_{h,2i}) \qquad B_{i,j} = a(\varphi^{(2)}_{h,2j}, \varphi^{(2)}_{h,2i-1})$$
$$C_{i,j} = a(\varphi^{(2)}_{h,2j-1}, \varphi^{(2)}_{h,2i}) \qquad D_{i,j} = a(\varphi^{(2)}_{h,2j-1}, \varphi^{(2)}_{h,2i-1})$$
$$c_i = \int_0^1 f(x)\varphi^{(2)}_{h,2i}(x)dx \qquad d_i = \int_0^1 f(x)\varphi^{(2)}_{h,2i-1}(x)dx$$
$$v_i = u^{(2)}_h(x^{(2)}_{h,2i}) \qquad w_i = u^{(2)}_h(x^{(2)}_{h,2i-1}).$$

Il est facile de vérifier que :

– la matrice A est tridiagonale,

– la matrice B est triangulaire supérieure et bidiagonale,

– la matrice C est triangulaire inférieure et bidiagonale,

– la matrice D est diagonale.

Les inconnues v et w sont solutions du système

$$\begin{cases} Av + Bw &= c \\ Cv + Dw &= d. \end{cases} \tag{4.11}$$

Les éléments diagonaux de la matrice D sont

$$D_{k,k} = a(\varphi^{(2)}_{h,2k-1}, \varphi^{(2)}_{h,2k-1}) = \varepsilon \int_{x^{(2)}_{h,2k-2}}^{x^{(2)}_{h,2k}} [\varphi^{(2)}_{h,2k-1}{}'(x)]^2 dx > 0.$$

Cette matrice D est donc inversible et on peut, dans la deuxième équation du système (4.11), exprimer w en fonction de v : $w = D^{-1}(d - Cv)$ et remplacer dans la première équation pour obtenir

$$(A - BD^{-1}C)v = c - BD^{-1}d. \tag{4.12}$$

Cette dernière équation permet de calculer le vecteur v c'est-à-dire les valeurs de la solution $P2$ aux extrémités des intervalles. Notons que la matrice $A - BD^{-1}C$ est tridiagonale alors que la matrice $A^{(2)}_h$ est pentadiagonale. Ce procédé qui consiste à isoler dans un système linéaire une partie des inconnues qui vérifient un système linéaire plus simple que le système d'origine est appelé *condensation*. Ce n'est rien d'autre qu'une élimination de Gauss des inconnues correspondant aux milieux des intervalles.

Exercice 4.7

1. Montrer que la matrice $A - BD^{-1}C$ est inversible.
2. Former les matrices A, B, C et D par extraction de lignes et colonnes de la matrice $A^{(2)}_h$ calculée à l'exercice 4.6.
3. On fixe $\lambda = 1$, $f = 1$ et $\varepsilon = 0.01$. Pour $n = 10$ et $n = 20$, résoudre le problème (4.12). Sur un même graphique

– tracer en trait continu la solution exacte calculée en 100 points de $[0, 1]$,

– représenter par des symboles la solution numérique, c'est-à-dire les valeurs du vecteur v.

4. Pour $n = 20$, comparer avec la méthode $P1$. À partir de quelle valeur n_0 la méthode $P1$ donne-t-elle une approximation de même qualité ? (on se contentera d'une appréciation visuelle des résultats).

La solution de cet exercice se trouve page 106.

La méthode $P2$ considérée uniquement aux extrémités des intervalles donne donc de bons résultats. Dans la section suivante, on justifie ces observations.

4.5.2 Analyse de la méthode stabilisée

Intéressons nous aux seules valeurs $u_h^{(2)}(x_{h,2k}^{(2)})$, c'est-à-dire les valeurs de $u_h^{(2)}$ sur la grille de la méthode $P1$ et montrons qu'elles peuvent être calculées par un schéma $P1$ légèrement modifié. Plus précisément, notant $A_h^{(1S)} = A - BD^{-1}C$, on a le résultat suivant

Proposition 4.1 *La matrice $A_h^{(1S)}$ est égale à la matrice $A_h^{(1)}$ dans laquelle ε a été remplacé par $\varepsilon' = \varepsilon + \lambda^2 h^2/(12\varepsilon)$.*

Ce résultat s'interprète comme un rajout de viscosité artificielle au schéma d'origine, ce qui rend la solution plus régulière (moins oscillante). Le reste de la section est consacré à la démonstration de la Proposition 4.1. Il s'agit de calculer explicitement $A_h^{(1S)}$. On notera $X = Tridiag(a, b, c; n, m)$ pour désigner une matrice X de taille $n \times m$ dont les seuls coefficients non nuls sont

$$X_{i-1,i} = a, X_{i,i} = b, X_{i,i+1} = c \quad \text{quand ces indices sont définis.}$$

Il s'agit donc de matrices tridiagonales, pas forcément carrées. Le lecteur aimant les calculs démontrera que

$$\begin{aligned}
A &= \frac{\varepsilon}{3h} Tridiag(1, 14, 1; n, n) + \frac{\lambda}{6} Tridiag(1, 0, -1; n, n),\\
B &= -\frac{8\varepsilon}{3h} Tridiag(0, 1, 1; n, n+1) - \frac{2\lambda}{3} Tridiag(0, 1, -1; n, n+1),\\
C &= -\frac{8\varepsilon}{3h} Tridiag(1, 1, 0; n+1, n) + \frac{2\lambda}{3} Tridiag(-1, 1, 0; n+1, n),\\
D &= \frac{16\varepsilon}{3h} Tridiag(0, 1, 0; n+1, n+1).
\end{aligned}$$

Puis, que (en notant $\sigma = \frac{16\varepsilon}{3h}$)

$$\begin{aligned} BC &= \left[\frac{\sigma}{2}\, Tridiag(0,1,1;n,n+1) + \frac{2\lambda}{3}\, Tridiag(0,1,-1;n,n+1)\right] \\ &\quad \times \left[\frac{\sigma}{2}\, Tridiag(1,1,0;n+1,n) - \frac{2\lambda}{3}\, Tridiag(-1,1,0;n+1,n)\right] \\ &= \frac{\sigma^2}{4}\, Tridiag(1,2,1;n,n) + \frac{\lambda\sigma}{3}\, Tridiag(2,0,-2;n,n) \\ &\quad -\frac{4}{9}\lambda^2\, Tridiag(-1,2,-1;n,n). \end{aligned}$$

Dans son élan, le lecteur calcule aussi

$$\begin{aligned} A_h^{(1S)} &= A - \frac{1}{\sigma} BC \\ &= \frac{\varepsilon}{3h}\, Tridiag(1,14,1;n,n) + \frac{\lambda}{6}\, Tridiag(1,0,-1;n,n) \\ &\quad -\frac{\sigma}{4}\, Tridiag(1,2,1;n,n) - \frac{\lambda}{3}\, Tridiag(2,0,-2;n,n) \\ &\quad +\frac{4}{9}\frac{\lambda^2}{\sigma}\, Tridiag(-1,2,-1;n,n) \\ &= Tridiag(\alpha,\beta,\gamma;n,n), \end{aligned}$$

avec

$$\begin{aligned} \alpha &= \frac{\varepsilon}{3h} + \frac{\lambda}{6} - \frac{\sigma}{4} - \frac{2\lambda}{3} - \frac{4}{9}\frac{\lambda^2}{\sigma} = -\frac{\varepsilon}{h} - \frac{\lambda}{2} - \frac{\lambda^2 h}{12\varepsilon} = -\frac{1}{h}\varepsilon' - \frac{\lambda}{2}, \\ \beta &= \frac{14\varepsilon}{3h} - \frac{\sigma}{2} + \frac{8}{9}\frac{\lambda^2}{\sigma} = 2\frac{\varepsilon}{h} + \frac{\lambda^2 h}{6\varepsilon} = \frac{2}{h}\varepsilon', \\ \gamma &= \frac{\varepsilon}{3h} - \frac{\lambda}{6} - \frac{\sigma}{4} + \frac{2\lambda}{3} - \frac{4}{9}\frac{\lambda^2}{\sigma} = -\frac{\varepsilon}{h} + \frac{\lambda}{2} - \frac{\lambda^2 h}{12\varepsilon} = -\frac{1}{h}\varepsilon' + \frac{\lambda}{2}. \end{aligned}$$

On a donc

$$A_h^{(1S)} = \frac{\varepsilon'}{h}\, Tridiag(-1,2,-1;n,n) + \frac{\lambda}{2}\, Tridiag(-1,0,1;n,n),$$

d'où le résultat annoncé puisque (voir l'exercice 4.3)

$$A_h^{(1)} = \frac{\varepsilon}{h}\, Tridiag(-1,2,-1;n,n) + \frac{\lambda}{2}\, Tridiag(-1,0,1;n,n).$$

4.6 CAS D'UN TERME SOURCE VARIABLE

On considère dans cette partie l'équation de convection diffusion (4.1) avec un terme source f non constant pour voir l'effet de ce terme sur l'existence d'une couche limite.

Exercice 4.8

1. On pose $\varepsilon = 0,01$, $\lambda = 1$ et $f(x) = \cos(a\pi x)$, $a \in \mathbb{R}$. Pour $a = 0$, on sait que la solution présente une couche limite au voisinage de 1. On suppose dans cet exercice que $a > 0$.

   Pour différentes valeurs de $a = 1, 2, 3, \ldots$, calculer numériquement (par l'un quelconque des schémas précédents) la solution de l'équation (4.1). On prendra n suffisamment grand pour que la solution numérique ne présente pas d'oscillations. Commenter les résultats. Même question pour $a = 3/2$.

2. Calcul de la solution exacte. On pose pour $x \in [0, 1]$

$$F_\theta(x) = \int_0^x e^{\theta z} \left[\int_0^z f(y) e^{-\theta y} dy \right] dz.$$

   a) Montrer que pour tous réels α et β, la fonction $\alpha + \beta e^{\theta x} - \dfrac{1}{\varepsilon} F_\theta$ est solution de l'équation différentielle (4.1).

   b) Déterminer α et β pour que $u = \alpha + \beta e^{\theta x} - \dfrac{1}{\varepsilon} F_\theta$ soit solution du problème (4.1) *i.e.* solution de l'équation différentielle et vérifie les conditions aux limites.

   c) Pour $f(x) = \cos(a\pi x)$ avec $a \in \mathbb{R}^*$, montrer que

$$\lim_{\varepsilon \to 0^+} u(x) = \frac{1}{\lambda a \pi} \sin(a\pi x).$$

3. Expliquer les résultats numériques observés à la question 1.

La solution de cet exercice se trouve page 106.

4.7 EN SAVOIR PLUS

Il existe plusieurs techniques de stabilisation des solutions numériques de problèmes de convection-diffusion. La technique étudiée ici est une stabilisation par "bulles" adaptée de l'article [Brezzi et Russo, 1994]. Le mot bulles (bubbles en anglais) vient du fait qu'on ajoute aux éléments finis $P1$ les fonctions $\varphi^{(2)}_{h,2k+1}$ qui ont une forme de bulle (voir la figure 4.4) ; la bulle $\varphi^{(2)}_{h,2k+1}$ est localisée dans l'intervalle I_k.

Les éléments finis en dimension deux d'espace sont étudiés au chapitre 11.

Pour les formulations variationnelles d'équations aux dérivées partielles ainsi que leurs discrétisations par éléments finis, nous renvoyons aux références [Bernardi, Maday et Rapetti, 2004], [Joly, 1990] et [Lucquin, 2004].

Pour l'implémentation sur ordinateur des méthodes, voir [Joly, 1990], [Danaila, Hecht et Pironneau, 2003] et [Lucquin et Pironneau, 1996].

4.8 SOLUTIONS ET PROGRAMMES

Les programmes nécessaires à la résolution des exercices de ce chapitre peuvent être téléchargés à partir du site web du livre.

Solution de l'exercice 4.1. Calcul de la solution exacte

1. Pour f constante, la solution u du problème (4.1) est

$$u(x) = \frac{f}{\lambda}(x - \frac{e^{\frac{\lambda x}{\varepsilon}} - 1}{e^{\frac{\lambda}{\varepsilon}} - 1}).$$

2. Posant $\theta = \frac{\lambda}{\varepsilon}$, on peut écrire

$$u(x) = \frac{f}{\varepsilon\theta}\left(x - \frac{e^{\theta x} - 1}{e^{\theta} - 1}\right).$$

et

$$\frac{\lambda}{f}u'(x) = 0 \Longleftrightarrow 1 - \frac{\theta e^{\theta x}}{e^{\theta} - 1} = 0 \Longleftrightarrow \theta e^{\theta x} = e^{\theta} - 1 \Longleftrightarrow x = \frac{1}{\theta}\ln\frac{e^{\theta} - 1}{\theta}.$$

On en déduit que $x_\theta = \frac{1}{\theta}\ln\frac{e^{\theta}-1}{\theta} = 1 + \frac{1}{\theta}\ln\frac{1-e^{-\theta}}{\theta} \in]0,1[$ et

$$u'(x) > 0 \Longleftrightarrow e^{\theta} - 1 - \theta e^{\theta x} > 0 \Longleftrightarrow \theta e^{\theta x} < \theta e^{\theta x_\theta} \Longleftrightarrow x < x_\theta.$$

On a $\lim\limits_{\theta\to+\infty} x_\theta = 1$ et $\lim\limits_{\theta\to-\infty} x_\theta = 0$.

3. On trouve

$$u(x_\theta) = \frac{f}{\lambda}\left(x_\theta - \frac{1}{\theta} + \frac{e^{-\theta x_\theta}}{\theta}\right), \quad \lim_{\varepsilon\to 0^+} u(x_\theta) = \frac{f}{\lambda}.$$

$$\lim_{\varepsilon\to 0^+}\lim_{x\to 1} u(x) = 0 \text{ et } \lim_{x\to 1}\lim_{\varepsilon\to 0^+} u(x) = \frac{f}{\lambda}.$$

La terme « couche limite » illustre bien le fait que la fonction a de fortes variations (elle passe de f/λ à 0) sur un petit intervalle $[x_\theta, 1]$ dont la longueur $1 - x_\theta = \frac{1}{\theta}\ln\frac{1-e^{-\theta}}{\theta}$ tend vers 0 quand θ tend vers $+\infty$.

4. La fonction `ConvecDiffSolExa` suivante calcule la solution exacte du problème

```
function y=ConvecDiffSolExa(e,lambda,fc,x)
  %solution du pb de convection diffusion
  %quand e, lambda et fc sont constants
    y=fc/lambda*(x-(1-exp(lambda*x/e))./(1-exp(lambda/e)));
```

La figure 4.6 montre les solutions pour $f = 1$, $\varepsilon \in \{1, 1/2, 10^{-1}, 10^{-2}\}$ et $\lambda = -1$ (figure de gauche) et $\lambda = -1$ (figure de droite). Quand ε tend vers 0 on

observe une zone autour de 1 ou -1 (suivant le signe de λ) où la solution varie "rapidement" d'une valeur proche de 1 à la valeur nulle ; c'est la couche limite.

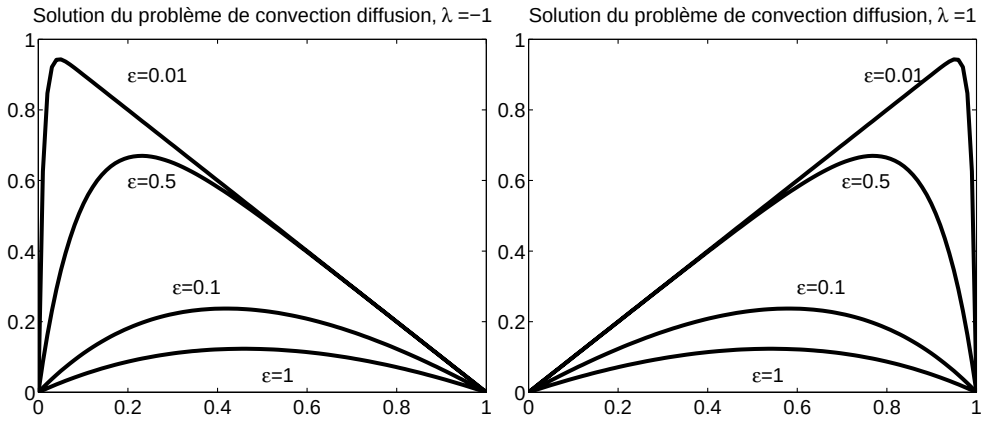


Figure 4.6 Solution du problème de convection-diffusion.

Solution de l'exercice 4.2

1. Il est clair que l'espace vectoriel $\mathcal{V}_1^h$ est de dimension n puisqu'il est isomorphe à $\mathbb{R}^n$. En effet, l'application qui à $u = (u_1, \ldots, u_n)^T \in \mathbb{R}^n$, associe la fonction $v \in \mathcal{V}_1^h$ définie par $v(x_i) = u_i$ est une bijection. Notant que $\varphi_{h,k}^{(1)}(x_j) = \delta_{j,k}$, on a

$$\left(\sum_{k=1}^{n} c_k \varphi_{h,k}^{(1)}(x) = 0, \forall x\right) \Longrightarrow \left(\sum_{k=1}^{n} c_k \varphi_{h,k}^{(1)}(x_j) = 0, \forall j\right) \Longrightarrow c_j = 0, \forall j,$$

   ce qui montre que la famille $(\varphi_{h,k}^{(1)})_{k=1}^n$ est libre. Comme $\mathcal{V}_1^h$ est un espace vectoriel de dimension n, la famille en question en est bien une base. On peut calculer les fonctions $\varphi_{h,k}^{(1)}$ ainsi que leurs dérivées

   – pour $x \notin I_{k-1} \cup I_k$, $\varphi_{h,k}^{(1)}(x) = \varphi_{h,k}^{(1)\prime}(x) = 0$,

   – pour $x \in I_{k-1}$, $\varphi_{h,k}^{(1)}(x) = (x - x_{k-1})/h$ et $\varphi_{h,k}^{(1)\prime}(x) = 1/h$,

   – pour $x \in I_k$, $\varphi_{h,k}^{(1)}(x) = (x_{k+1} - x)/h$ et $\varphi_{h,k}^{(1)\prime}(x) = -1/h$.

2. Puisque $(\varphi_{h,j}^{(1)})_{j=1}^n$ forme une base de $\mathcal{V}_1^h$, on peut remplacer dans la relation(4.5) "$v_h \in \mathcal{V}_1^h$" par "v_h décrivant la base $(\varphi_{h,j}^{(1)})_{j=1}^n$".

3. Exploitant toujours les relations $\varphi_{h,k}^{(1)}(x_j) = \delta_{j,k}$, on a

$$u_h^{(1)}(x_j) = \sum_{k=1}^{n} \alpha_k \varphi_{h,k}^{(1)}(x_j) = \alpha_j.$$

En remplaçant u_h par son développement dans la base des $\varphi_{h,k}^{(1)}$, on obtient

$$\sum_{k=1}^{n} a(\varphi_{h,k}^{(1)}, \varphi_{h,j}^{(1)})\alpha_k = \int_0^1 f\varphi_{h,j}dx, \quad \forall j = 1, \ldots, n.$$

Soient $A_h^{(1)}$ la matrice carrée $n \times n$ et $b_h^{(1)}$ le vecteur de $\mathbb{R}^n$ définis par

$$(A_h^{(1)})_{j,k} = a(\varphi_{h,k}^{(1)}, \varphi_{h,j}^{(1)}), \quad (b_h^{(1)})_j = \int_0^1 f\varphi_{h,j}^{(1)}dx.$$

Le vecteur $\tilde{u}_h = (\alpha_1, \ldots, \alpha_n)^T$ est solution du système linéaire $A_h^{(1)}\tilde{u}_h^{(1)} = b_h^{(1)}$. On peut écrire $A_h^{(1)} = \varepsilon B_h^{(1)} + \lambda C_h^{(1)}$ avec

$$(B_h^{(1)})_{j,k} = \int_0^1 \varphi_{h,k}^{(1)\prime}\varphi_{h,j}^{(1)\prime}dx, \quad \text{et } (C_h^{(1)})_{j,k} = \int_0^1 \varphi_{h,k}^{(1)\prime}\varphi_{h,j}^{(1)}dx.$$

Il est clair que $(B_h^{(1)})_{j,k} = (B_h^{(1)})_{k,j}$ et qu'en intégrant par parties et notant que les fonctions de base sont nulles en 0 et 1 que $(C_h^{(1)})_{j,k} = -(C_h^{(1)})_{k,j}$. Le caractère tridiagonal des deux matrices découle du fait que les supports de deux fonctions $\varphi_{h,j}^{(1)}$ et $\varphi_{h,k}^{(1)}$ sont disjoints pour $|j-k| > 1$.

4. Soit $x \in \mathbb{R}^n$, de composantes $(x_k)_{k=1}^n$ on a

$$\begin{aligned}
\langle B_h^{(1)}x, x\rangle &= \sum_{k=1}^{n}(B_h^{(1)}x)_k x_k = \sum_{k=1}^{n}\sum_{j=1}^{n}(B_h^{(1)})_{k,j}x_j x_k \\
&= \sum_{k=1}^{n} x_k \sum_{j=1}^{n} x_j \int_0^1 \varphi_{h,k}^{(1)\prime}(x)\varphi_{h,j}^{(1)\prime}(x)dx \\
&= \int_0^1 \left(\sum_{k=1}^{n} x_k \varphi_{h,k}^{(1)\prime}(x)\right)^2 dx \geqslant 0.
\end{aligned}$$

De plus $\langle B_h^{(1)}x, x\rangle = 0$ implique que $\sum_{k=1}^n x_k\varphi_{h,k}^{(1)\prime}(x) = 0$ pour tout $x \in [0,1]$, donc que les x_k sont nuls. La matrice symétrique $B_h^{(1)}$ est par conséquent définie positive.

5. Puisque la matrice $C_h^{(1)}$ est antisymétrique, on a

$$\langle C_h^{(1)}x, x\rangle = \langle x, {C_h^{(1)}}^T x\rangle = -\langle x, C_h^{(1)}x\rangle = -\langle C_h^{(1)}x, x\rangle,$$

soit $\langle C_h^{(1)}x, x\rangle = 0$. D'où le résultat.

Solution de l'exercice 4.3. Calcul numérique de la solution $P1$

1. Calcul de A_h. On note que les supports de deux fonctions $\varphi^{(1)}_{h,j}$ et $\varphi^{(1)}_{h,k}$ assez « éloignées » sont disjoints. Plus précisément, notant $b_{k,j} = \int_0^1 \varphi^{(1)\prime}_{h,k}\varphi^{(1)\prime}_{h,j}$ et $c_{k,j} = \int_0^1 \varphi^{(1)\prime}_{h,k}\varphi_{h,j}$, on a :

– pour $|k-j| > 1$, $b_{k,j} = c_{k,j} = 0$,

– pour $k = j$,

$$b_{k,k} = \int_0^1 (\varphi^{(1)\prime}_{h,k})^2 = \int_{I_{k-1}} (\varphi^{(1)\prime}_{h,k})^2 + \int_{I_k} (\varphi^{(1)\prime}_{h,k})^2 = \frac{2}{h},$$

$$c_{k,k} = \int_0^1 \varphi^{(1)\prime}_{h,k}\varphi_{h,k} = \int_{I_{k-1}} \varphi^{(1)\prime}_{h,k}\varphi_{h,k} + \int_{I_k} \varphi^{(1)\prime}_{h,k}\varphi_{h,k} = 0,$$

– pour $k = j+1$,

$$b_{j+1,j} = b_{j,j+1} = \int_0^1 \varphi^{(1)\prime}_{h,j+1}\varphi^{(1)\prime}_{h,j} = -\frac{1}{h},$$

$$c_{j+1,j} = -c_{j,j+1} = \int_0^1 \varphi^{(1)\prime}_{h,j+1}\varphi_{h,j} = -\frac{1}{2}.$$

On a donc bien la décomposition $A_h = \varepsilon B_h + \lambda C_h$.

2. Voir la fonction `ConvecDiffAP1.m`.

3. Utilisant la méthode des trapèzes, on écrit

$$(b^{(1)}_h)_k = \int_0^1 f\varphi^{(1)}_{h,k}dx = \int_{x^{(1)}_{k-1}}^{x^{(1)}_k} f\varphi^{(1)}_{h,k}dx + \int_{x^{(1)}_k}^{x^{(1)}_{k+1}} f\varphi^{(1)}_{h,k}dx$$

$$\simeq \frac{h}{2}\left[f(x^{(1)}_{k-1})\varphi^{(1)}_{h,k}(x^{(1)}_{k-1}) + 2f(x^{(1)}_k)\varphi^{(1)}_{h,k}(x^{(1)}_k) + f(x^{(1)}_{k+1})\varphi^{(1)}_{h,k}(x^{(1)}_{k+1})\right] = hf(x^{(1)}_k).$$

D'où la fonction `ConvecDiffbP1`.

4. Le script suivant génére la figure 4.7. Pour les valeurs choisies de λ et ε, on obtient une bonne approximation. Ce script est disponible sur le site web du livre sous le nom de `ConvecDiffscript1W.m`.

```
eps=0.1;lambda=1;                    %paramètres physiques
f = inline('ones(size(x))');         %second membre de l'équation
n=10;
A=ConvecDiffAP1(eps,lambda,n);       %matrice du système linéaire
b=ConvecDiffbP1(n,f);                %second membre du systeme linéaire
u=A\ b;                              %solution "éléments finis"
u=[0;u;0];              %on complète u par les conditions aux bords
x=(0:n+1)/(n+1);                      %grille de calcul
uexa=ConvecDiffSolExa(eps,lambda,1,x); %solution exacte
                                       %calculée sur x
plot(x,uexa,x,u,'+-r')
```

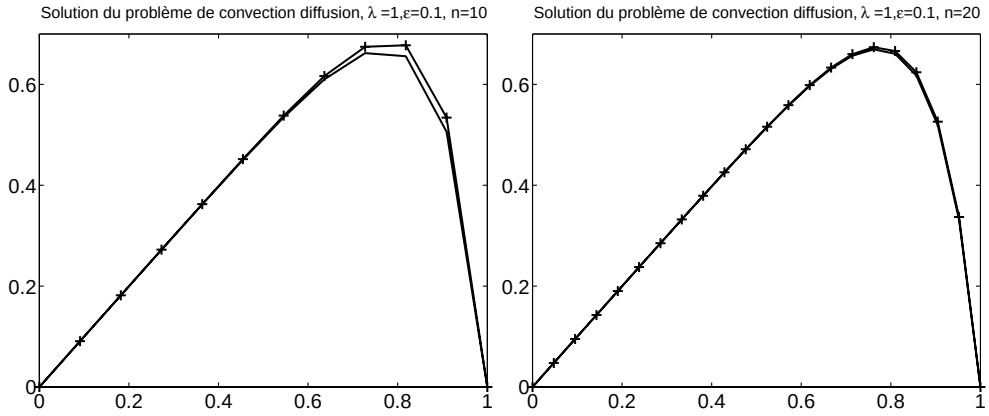


Figure 4.7 Approximation du problème de convection-diffusion (éléments finis $P1$), $\varepsilon = 0.1$, $\lambda = 1$, $n = 10$ (à gauche) et $n = 20$ (à droite).

5. Étude de l'erreur. La figure 4.8 est obtenue par le script ci-dessous ; on y observe une belle droite de pente égale (environ) à -2. On en déduit que $s_1 = 2$ et que si on raffine le maillage en passant de h à $h/2$, l'erreur est divisée par $2^{s_1} = 4$.

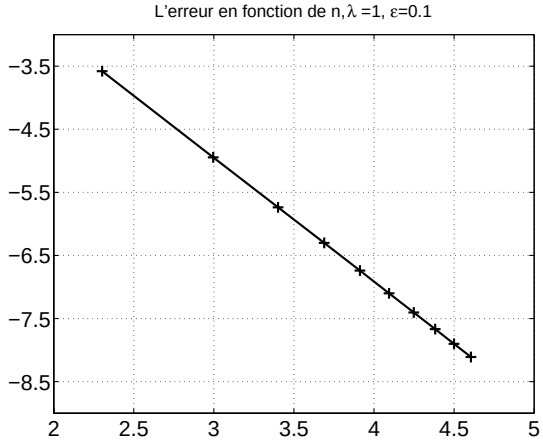


Figure 4.8 Approximation du problème de convection-diffusion (éléments finis $P1$), Logarithme de l'erreur en fonction du logarithme de n.

```
erreur=[];N=[];
for n=10:10:100
   A=ConvecDiffAP1(eps,lambda,n);  %matrice du systéme linèaire
   b=ConvecDiffbP1(n,f);           %second membre du système linéaire
   u=A\b;                          %solution "éléments finis"
   u=[0;u;0];            %on complète u par les conditions aux bords
   x=(0:n+1)'/(n+1);                %grille de calcul
```

```
  uexa=ConvecDiffSolExa(eps,lambda,1,x);  %solution exacte
                                           %calculée sur x
  N=[N;n];erreur=[erreur; norm(uexa-u,'inf')];
end
plot(log(N),log(erreur),'+-');
```

Voir le script `ConvecDiffscript2W.m`.

Solution de l'exercice 4.4

Exemple de script : on fixe ε et fait varier n. On note que l'approximation est bonne quand le nombre de Peclet est inférieur à 1.

```
eps=0.01;lambda=1;                    %paramètres physiques
f = inline('ones(size(x))');          %second membre de l'équation
oui=1;
while oui
   n=input('entrer la valeur de n : ');
   A=ConvecDiffAP1(eps,lambda,n);      %matrice du système linéaire
   b=ConvecDiffbP1(n,f);             %second membre du système linéaire
   u=A\ b;                           %solution "éléments finis"
   u=[0;u;0];                        %on complete u par les conditions
aux bords
   x=(0:n+1)/(n+1);                             %grille de calcul
   uexa=ConvecDiffSolExa(eps,lambda,1,x);       %solution exacte
                                                %calculée sur x
   plot(x,uexa,x,u,'+-r')
   Peclet=abs(lambda)/2/eps/(n+1)
   oui=input('on continue? oui=1, non=0 ')
end
```

Voir le script `ConvecDiffscript3W.m`.

Solution de l'exercice 4.5

Dans la méthode $P1$, les matrices $B_h^{(1)}$ et $C_h^{(1)}$ sont tridiagonales. Dans la méthode $P2$ les supports des fonctions de base sont plus étalés et les matrices $B_h^{(2)}$ et $C_h^{(2)}$ sont pentadiagonales. Les mêmes démonstrations que dans le cas $P1$ montrent que la matrice $B_h^{(2)}$ est symétrique et définie positive, la matrice $C_h^{(2)}$ est antisymétrique et la matrice $A_h^{(2)}$ est inversible.

Solution de l'exercice 4.6

Les dérivées des fonctions de base sont

$$\varphi_{h,2k+1}^{(2)\,\prime}(x) = \begin{cases} -8(x - x_{2k+1}^{(2)})/h^2 & \text{pour } x \in I_k, \\ 0 & \text{sinon.} \end{cases}$$

et

$$\varphi_{h,2k}^{(2)\prime}(x) = \begin{cases} 4(x - x_{2k+3/2}^{(2)})/h^2 & \text{pour } x \in I_k, \\ 4(x - x_{2k-1/2}^{(2)})/h^2 & \text{pour } x \in I_{k-1}, \\ 0 & \text{sinon.} \end{cases}$$

1. a) Calcul de $B_h^{(2)}$. La matrice est symétrique, on calcule sa partie triangulaire supérieure.

– Lignes d'indices impairs.

- $(B_h^{(2)})_{2k+1,2k+1} = \int_0^1 [\varphi_{h,2k+1}^{(2)\prime}(x)]^2 dx = \int_{x_{2k}^{(2)}}^{x_{2k+2}^{(2)}} [\frac{8}{h^2}(x - x_{2k+1}^{(2)})]^2 dx = \frac{16}{3}\frac{1}{h}$
- $(B_h^{(2)})_{2k+1,2k+2} = -\frac{8}{3}\frac{1}{h}$
- $(B_h^{(2)})_{2k+1,m} = 0, \quad \forall m \geqslant 2k+3$

– Lignes d'indices pairs.

- $(B_h^{(2)})_{2k,2k} = \frac{14}{3}\frac{1}{h}$
- $(B_h^{(2)})_{2k,2k+1} = -\frac{8}{3}\frac{1}{h}$
- $(B_h^{(2)})_{2k,2k+2} = \frac{1}{3}\frac{1}{h}$
- $(B_h^{(2)})_{2k,m} = 0, \quad \forall m \geqslant 2k+3$

La partie triangulaire supérieure de la matrice $B_h^{(2)}$ est donc

$$\frac{1}{3h}\begin{pmatrix} 16 & -8 & 0 & 0 & 0 & 0 & 0 \\ & 14 & -8 & 1 & 0 & 0 & 0 \\ & & 16 & -8 & 0 & 0 & 0 \\ & & & 14 & -8 & 1 & 0 \\ & & & & \ddots & \ddots & \ddots \\ & & & & & \ddots & \ddots \\ & & & & & & \ddots \end{pmatrix}$$

b) Calcul de $C_h^{(2)}$. La matrice est antisymétrique, on calcule sa partie triangulaire supérieure.

– Lignes d'indices impairs.

- $(C_h^{(2)})_{2k+1,2k+1} = 0$
- $(C_h^{(2)})_{2k+1,2k+2} = \frac{2}{3}$
- $(C_h^{(2)})_{2k+1,m} = 0, \quad \forall m \geqslant 2k+3$

– Lignes d'indices pairs.

- $(C_h^{(2)})_{2k,2k} = 0$
- $(C_h^{(2)})_{2k,2k+1} = \frac{2}{3}$

- $(C_h^{(2)})_{2k,2k+2} = -\frac{1}{6}$
- $(C_h^{(2)})_{2k,m} = 0, \quad \forall m \geqslant 2k+3$

La partie triangulaire supérieure de la matrice $C_h^{(2)}$ est donc

$$\frac{1}{3}\begin{pmatrix} 0 & -2 & 0 & 0 & 0 & 0 & 0 \\ & 0 & -2 & 1/2 & 0 & 0 & 0 \\ & & 0 & -2 & 0 & 0 & 0 \\ & & & 0 & -2 & 1/2 & 0 \\ & & & & \ddots & \ddots & \ddots \\ & & & & & \ddots & \ddots \\ & & & & & & \ddots \end{pmatrix}.$$

2. Voir la fonction `ConvecDiffAP2.m`.
3. Utilisant la méthode de Simpson, on écrit

$$\begin{aligned}(b_h^{(2)})_{2k+1} &= \int_0^1 f\varphi_{h,2k+1}^{(2)}dx = \int_{x_{2k}^{(2)}}^{x_{2k+2}^{(2)}} f\varphi_{h,2k+1}^{(2)}dx \\ &\simeq \frac{h}{6}\left[f(x_{2k}^{(2)})\varphi_{h,2k+1}^{(2)}(x_{2k}^{(2)}) + 2f(x_{2k+1}^{(2)})\varphi_{h,k}^{(2)}(x_{2k+1}^{(1)}) + f(x_{2k+2}^{(2)})\varphi_{h,2k+1}^{(2)}(x_{2k+2}^{(2)})\right] \\ &= \frac{2}{3}hf(x_{2k+1}^{(2)})\end{aligned}$$

De même $(b_h^{(2)})_{2k} \simeq \frac{1}{3}hf(x_{2k}^{(2)})$. D'où la fonction`ConvecDiffbP2`.

4. Étude de l'erreur. Comme à l'exercice 4.3, on trace sur la figure 4.9 le logarithme de l'erreur en fonction du logarithme de n. La courbe obtenue est une droite de pente égale (environ) à -3.3. L'erreur décroit donc plus vite que dans la méthode d'éléments finis $P1$.

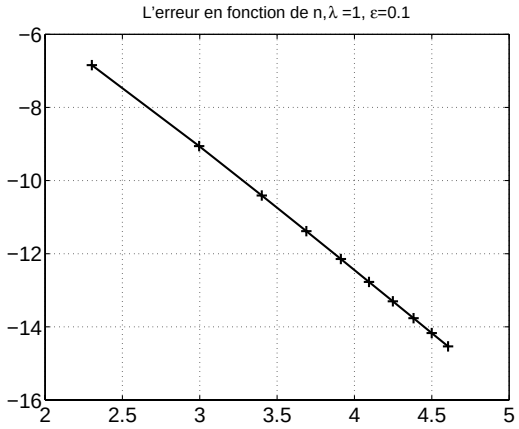


Figure 4.9 Approximation du problème de convection-diffusion (éléments finis $P2$), Logarithme de l'erreur en fonction du logarithme de n.

Note. Pour comparer correctement les méthodes $P1$ et $P2$, il faut considérer la quantité suivante

$$\int_0^1 |u_h'(x) - u'(x)|^2 dx = \sum_{\text{intervalles } I_k} \int_{I_k} |u_h'(x) - u'(x)|^2 dx.$$

Voir à ce sujet, les références données à la fin de ce chapitre.

Solution de l'exercice 4.7.

1. Soit x un vecteur non nul de $\mathbb{R}^n$ tel que $(A - BD^{-1}C)x = 0$. Pour le vecteur non nul $y = (x^T, -(D^{-1}Cx)^T)^T \in \mathbb{R}^{2n+1}$, on a par construction $\widehat{A}_h^{(2)} y = 0$ or la matrice $\widehat{A}^{(2)}$ est inversible. D'où une contradiction. La matrice carrée $A - BD^{-1}C$ est donc injective et par conséquent inversible.

2.

```
A=ConvecDiffAP2(eps,lambda,n);
a=A(2:2:2*n+1,2:2:2*n+1);b=A(2:2:2*n+1,1:2:2*n+1);
c=A(1:2:2*n+1,2:2:2*n+1);d=A(1:2:2*n+1,1:2:2*n+1);
```

3. Le script suivant est disponible sur le site Web du livre sous le nom de `ConvecDiffscript4W.m`.

```
n=10;eps=0.01;lambda=1;
f = inline(ones(size(x))');
sm=ConvecDiffbP2(n,f);
nsm=sm(2:2:2*n+1)-b*inv(d)*sm(1:2:2*n+1);
u=(a-b*inv(d)*c)\ nsm;                          %calcul de v
x=linspace(0,1,100);
uexa=ConvecDiffSolExa(eps,lambda,1,x);
plot(x,uexa);hold on
plot((1:n)/(n+1),u,'+');hold off;
```

Pour $n = 10$ et $n = 20$, voir la figure 4.10.

4. On présente sur la figure 4.11 la solution numérique stabilisée avec $n = 20$ et les solutions $P1$ pour $n \in \{20, 40, 60, 80\}$

Solution de l'exercice 4.8.

1. Pour le même jeu de paramètre et une fonction f constante, on avait observé une couche limite, il n'en est rien ici. On représente les résultats sur la figure 4.12. On y observe que le terme source f transfère son aspect oscillant à la solution u.

   Voici le script générant la figure 4.12. Ce script est disponible sur le site Web du livre sous le nom de `ConvecDiffscript5W.m`.

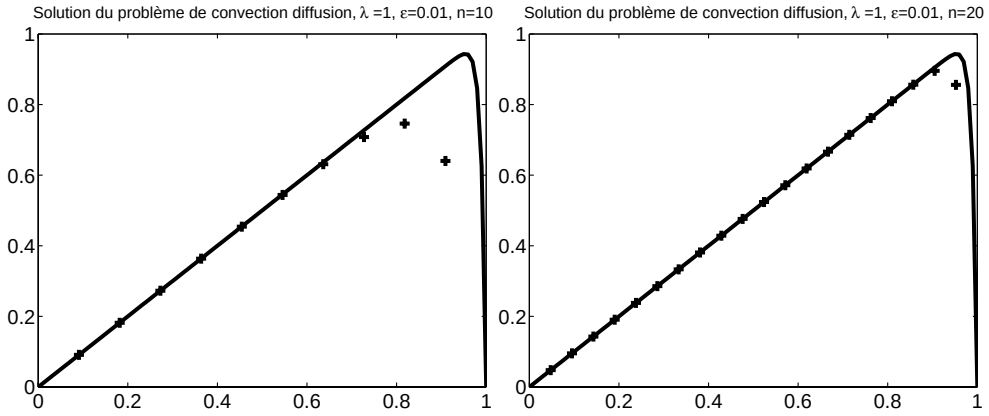


Figure 4.10 Solution stabilisée du problème de convection-diffusion : $\lambda = 1$, $\varepsilon = 0.01$ et $n = 10$ (à gauche) et $n = 20$ (à droite).

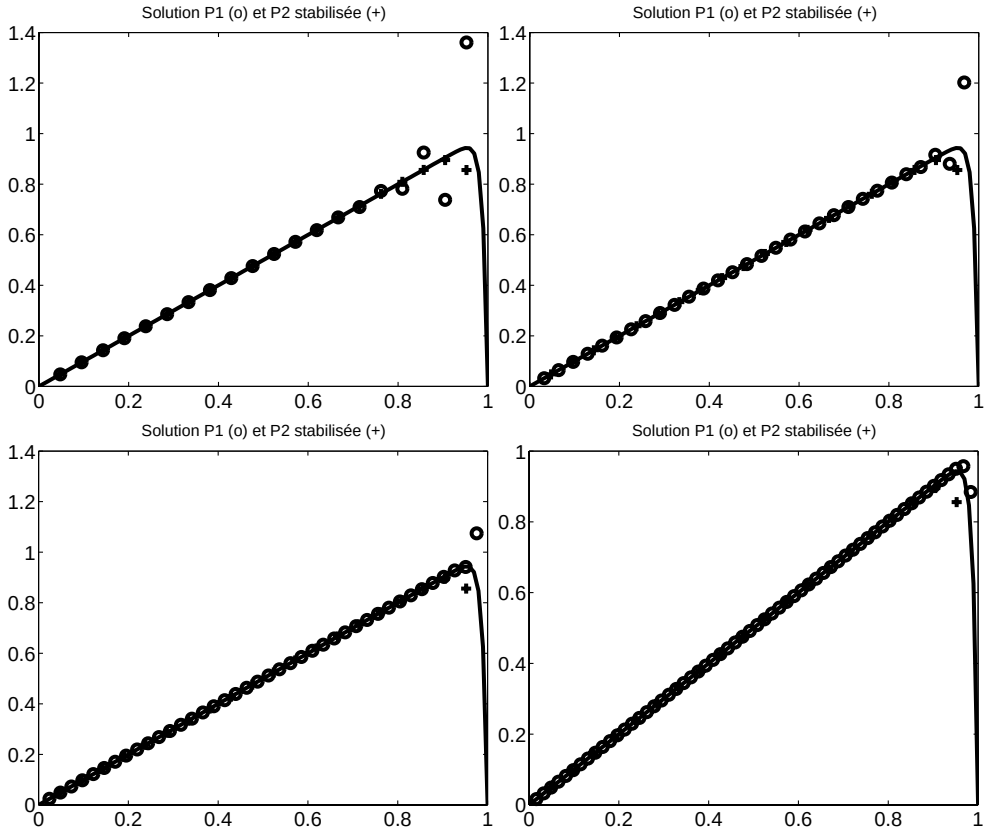


Figure 4.11 Solution stabilisée avec $n = 20$ et solutions $P1$ avec $n = 20$ (en haut, à gauche), $n = 30$ (en haut, à droite), $n = 40$ (en bas, à gauche), $n = 60$ (en bas, à droite).

```
n=100;lambda=1;eps=0.01;
x=(0:(n+1))'/(n+1);
A=ConvecDiffAP1(eps,lambda,n);
X=[];Y=[];
h=1/(n+1);tab=(1:n)'*h;
for af=1:5
   b=h*cos(af*pi*tab);
   y=A\ b;y=[0; y; 0];
   X=[X x];Y=[Y y];
end;
plot(X,Y);
```

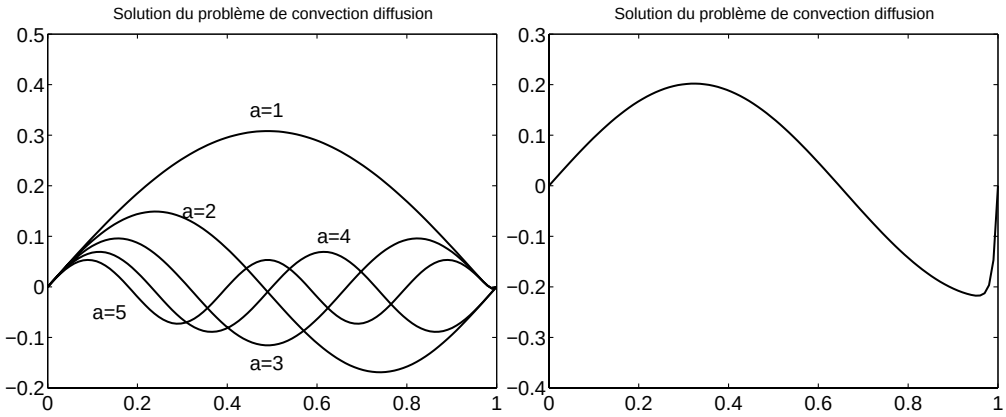


Figure 4.12 Solution de l'équation de convection-diffusion. À gauche : $f(x) = \cos(a\pi x)$, pour $a \in \{1, 2, 3, 4, 5\}$. À droite : $f(x) = \cos(\frac{3}{2}\pi x)$.

Pour $a = 3/2$, on observe cette fois une couche limite, voir la figure 4.12. On explique ce comportement à la question suivante.

2. Calcul de la solution exacte.

a) On vérifie facilement que $\alpha + \beta e^{\theta x} - \frac{1}{\varepsilon} F_\theta$ est bien solution de l'équation différentielle (4.1).

b) Les conditions $u(0) = 0$ et $u(1) = 0$ permettent de déterminer α et β

$$\alpha = -\beta = \frac{1}{\varepsilon} \frac{F_\theta(1)}{1 - e^\theta}$$

et donc la solution est

$$u(x) = \frac{1}{\varepsilon(1 - e^\theta)} [(1 - e^{\theta x}) F_\theta(1) - (1 - e^\theta) F_\theta(x)]. \tag{4.13}$$

c) $f(x) = \cos(a\pi x)$. En posant $I = \int_0^z \cos(a\pi y) e^{-\theta y} dy$, $J = \int_0^z \sin(a\pi y) e^{-\theta y} dy$ et calculant la partie réelle de $I + iJ$, on obtient

$$\int_0^z f(y) e^{-\theta y} dy = \frac{a\pi e^{-\theta z} \sin(a\pi z) + \theta - \theta e^{-\theta z} \cos(a\pi z)}{\theta^2 + \pi^2 a^2}.$$

On en déduit que

$$(\theta^2 + \pi^2 a^2) F_\theta(x) = e^{\theta x} - \cos(a\pi x) - \frac{\theta \sin(a\pi x)}{a\pi}.$$

La solution du problème (4.1) est la somme de deux termes $-\frac{1}{\varepsilon} F_\theta(x)$ et $\alpha + \beta e^{\theta x}$.

– Le premier terme peut se décomposer en trois termes :
  - $\frac{1}{\varepsilon} \frac{\theta}{(\theta^2+\pi^2 a^2)} \frac{\sin(a\pi x)}{a\pi}$, qui tend vers $\frac{\lambda}{a\pi} \sin(a\pi x)$ quand ε tend vers 0,
  - $\frac{1}{\varepsilon} \frac{\cos(a\pi x)}{(\theta^2+\pi^2 a^2)}$, qui tend vers 0,
  - et $-\frac{1}{\varepsilon} \frac{e^{\theta x}}{\theta^2+\pi^2 a^2}$, qu'on notera γ_1.

– Le seul terme de $\alpha + \beta e^{\theta x}$ qui ne tend pas vers 0 est $\gamma_2 = \frac{1}{\varepsilon} \frac{e^\theta}{\theta^2+\pi^2 a^2} \frac{1-e^{\theta x}}{1-e^\theta}$.
Comme la somme $\gamma_1 + \gamma_2$ tend vers 0, on en déduit que

$$\lim_{\varepsilon \to 0^+} u(x) = \frac{1}{\lambda a\pi} \sin(a\pi x).$$

3. La fonction u est continue et vérifie la condition à la limite $u(1) = 0$, donc

$$\lim_{\varepsilon \to 0^+} \lim_{x \to 1} u(x) = \lim_{\varepsilon \to 0^+} u(1) = 0.$$

D'autre part, $\lim\limits_{x \to 1} \lim\limits_{\varepsilon \to 0^+} u(x) = \dfrac{1}{\lambda a\pi} \sin(a\pi)$. On en déduit que pour a entier, Il n'y a pas de couche limite au voisinage de $x = 1$ puisque $\lim\limits_{x \to 1} \lim\limits_{\varepsilon \to 0^+} u(x) = \lim\limits_{\varepsilon \to 0^+} \lim\limits_{x \to 1} u(x) = 0$.
Mais pour les valeurs non entières de a entier, il y a une couche limite car alors $\lim\limits_{x \to 1} \lim\limits_{\varepsilon \to 0^+} u(x) \neq 0$.

BIBLIOGRAPHIE

[Bernardi, Maday et Rapetti, 2004] C. BERNARDI, Y. MADAY, F. RAPETTI : *Discrétisations variationnelles de problèmes aux limites elliptiques, collection S.M.A.I. Mathématiques et Applications*, vol. 45, Springer Verlag, Paris, 2004.

[Brezzi et Russo, 1994] F. BREZZI, A. RUSSO : *Choosing bubbles for advection-diffusion problems.*, Math. Models and Meth. in Appl. Sci., vol 4, n. 4, 1994.

[Danaila, Hecht et Pironneau, 2003] I. DANAILA, F. HECHT, O. PIRONNEAU : *Simulation numérique en C++*, Dunod, Paris, 2003.

[Joly, 1990] P. JOLY : *Mise en œuvre de la méthode des éléments finis, collection S.M.A.I. Mathématiques et Applications*, Ellipses, Paris, 1990.

[Lucquin, 2004] B. LUCQUIN : *Équations aux dérivées partielles et leurs approximations*, Ellipses, Paris, 2004.

[Lucquin et Pironneau, 1996] B. LUCQUIN, O. PIRONNEAU : *Introduction au calcul scientifique*, Masson, Paris, 1996.

Projet **5**

Une méthode spectrale pour la résolution d'une équation différentielle

Fiche du projet

Difficulté :	2
Notions développées :	Méthode spectrale, approximation polynomiale, quadrature de Gauss, polynômes orthogonaux de Legendre
Domaines d'application :	Formulation variationnelle, discrétisation du laplacien

Les méthodes spectrales sont des techniques d'approximation des solutions d'équations aux dérivées partielles. Elles consistent à rechercher la solution approchée dans un espace de polynômes. La précision de ces méthodes n'est limitée que par la régularité de la fonction à approcher, à l'opposé des approximations de type différences finies ou éléments finis basées sur une discrétisation du domaine de définition de la solution. La technique pour calculer la solution approchée repose essentiellement sur la formulation variationnelle du problème continu. L'approximation consiste à remplacer l'espace de fonctions test par un espace de polynômes et à évaluer les intégrales au moyen de formules de quadrature appropriées.

Le projet propose de mettre en oeuvre une méthode spectrale pour résoudre numériquement le problème aux limites posé sur l'intervalle $\Omega =]-1,1[$:

$$\begin{cases} -u'' + cu = f, \\ u(-1) = 0, \\ u(1) = 0, \end{cases} \tag{5.1}$$

avec $f \in L^2(\Omega)$ et c un réel positif ou nul.

La première partie du projet consiste à rappeler quelques propriétés des polynômes de Legendre qui serviront à construire une base de l'espace d'approximation de la solution. Le calcul par récurrence des polynômes est programmé. Dans la deuxième partie on définit et on met en oeuvre l'approximation d'une fonction par sa série de Legendre tronquée, ce qui nécessite également l'introduction d'une méthode de quadrature pour approcher numériquement les intégrales définissant les coefficients de cette série. Enfin dans la troisième partie du projet, on met en oeuvre la résolution numérique de l'équation différentielle (5.1) par la méthode spectrale.

5.1 QUELQUES PROPRIÉTÉS DES POLYNÔMES DE LEGENDRE

On considère l'espace vectoriel $\mathbb{P}_n$ des polynômes de degré inférieur ou égal à n, et on note $(L_n)_{0 \leqslant n}$ la famille des polynômes de Legendre définis par

$$\begin{cases} L_0(x) = 1, \\ L_1(x) = x, \\ (n+1)L_{n+1}(x) = (2n+1)xL_n(x) - nL_{n-1}(x), \quad \text{pour} n \geqslant 1 \end{cases} \tag{5.2}$$

Ces polynômes vérifient également l'équation différentielle

$$[(1-x^2)L_n'(x)]' + n(n+1)L_n(x) = 0, \qquad n \geqslant 0. \tag{5.3}$$

et les relations $L_n(\pm 1) = (\pm 1)^n$.

Les polynômes de Legendre forment une base orthogonale pour le produit scalaire usuel sur $]-1,1[$

$$\langle f, g \rangle = \int_{-1}^{1} f(x)g(x)dx.$$

Plus précisément, pour tous entiers n et m, on a

$$\langle L_n, L_m \rangle = \frac{2}{2n+1}\delta_{n,m}. \tag{5.4}$$

Exercice 5.1

1. Écrire une fonction `y=CombLinLeg(x,c)` calculant les valeurs d'une combinaison linéaire donnée de polynômes de Legendre (5.5) en des points

constituant les abscisses d'un vecteur `x`.

$$y = \sum_{k=1}^{p} c_k L_{k-1}(x) \tag{5.5}$$

2. Écrire un script appelant la fonction `CombLinLeg`, avec `x` qui discrétise l'intervalle $[-1, 1]$, suffisamment finement pour une bonne représentation graphique, et `c` contenant les coefficients correspondants à la combinaison linéaire $L_0 - 2L_1 + 3L_5$.

Tracer `y` en fonction de `x` comme sur la figure 5.1.

La solution de cet exercice se trouve en page 119.

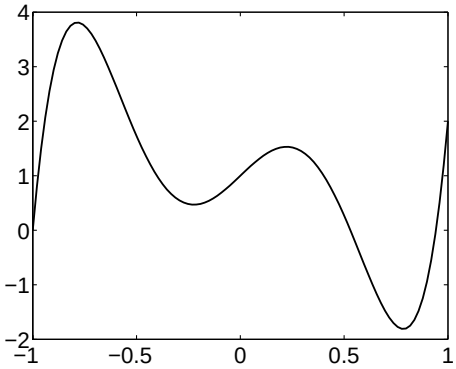


Figure 5.1 Combinaison linéaire $L_0 - 2L_1 + 3L_5$.

5.2 INTÉGRATION NUMÉRIQUE PAR QUADRATURE DE GAUSS

D'une façon générale, les méthodes d'intégration numérique (voir [Delabrière et Postel, 2004] et [Crouzeix et Mignot, 1989]) permettent de calculer une valeur approchée de l'intégrale d'une fonction quand on ne connaît pas sa primitive mais qu'on peut évaluer les valeurs de la fonction elle-même en un certain nombre de points.

La méthode d'intégration numérique par quadrature de Gauss, repose sur le résultat d'approximation suivant :

$$\int_{-1}^{1} \varphi(x)dx = \sum_{i=1}^{s} \varphi(x_i)\omega_i + R_s(\varphi)$$

où les x_i sont les zéros du polynôme de Legendre L_s, et où les $poids$ ω_i sont donnés par la relation

$$\omega_i = \frac{2}{(1 - x_i^2)[L_s'(x_i)]^2}. \tag{5.6}$$

Le reste

$$R_s(\varphi) = \frac{2^{2s+1}(s!)^4}{(2s+1)[(2s)!]^3}\varphi^{(2s)}(\xi), \quad \text{avec} \quad \xi \in]-1, 1[$$

est nul pour les polynômes de $\mathbb{P}_{2s-1}$. Pour ces fonctions, la formule de quadrature

$$\int_{-1}^{1} \varphi(x)dx \simeq \sum_{i=1}^{s} \varphi(x_i)\omega_i \tag{5.7}$$

est exacte.

Pour utiliser numériquement cette formule, il faut tout d'abord calculer les poids ω_i et les points x_i d'intégration. On propose pour cela la méthode ci-dessous.

Les relations (5.2) sur les polynômes L_j pour $j = 0, \ldots, s$ peuvent s'écrire sous forme matricielle $Mu = xu + v$, en posant $a_j = j/(2j+1)$, $b_j = (j+1)/(2j+1)$, et

$$M = \begin{pmatrix} 0 & 1 & & & \\ a_1 & 0 & b_1 & & \\ & \ddots & \ddots & \ddots & \\ & & a_{s-2} & 0 & b_{s-2} \\ & & & a_{s-1} & 0 \end{pmatrix}, \quad v = b_{s-1}L_s(x) \begin{pmatrix} 0 \\ \vdots \\ \vdots \\ 0 \\ 1 \end{pmatrix}, \quad u = \begin{pmatrix} L_0(x) \\ \vdots \\ \vdots \\ \vdots \\ L_{s-1}(x) \end{pmatrix}.$$

Pour x racine de L_s, on a $v = 0$ et le système linéaire devient $Mu = xu$. Donc les valeurs propres de la matrice tridiagonale M sont les racines de L_s.

Pour calculer les poids ω_i, donnés par la formule (5.6) on combine la relation de récurrence (5.2) avec la relation suivante

$$(1 - x^2)L_s'(x) = -sxL_s(x) + sL_{s-1}(x), \quad s \geqslant 1. \tag{5.8}$$

Comme $L_s(x_i) = 0$, on obtient finalement

$$\omega_i = \frac{2(1 - x_i^2)}{(sL_{s-1}(x_i))^2}$$

et on calcule la valeur $L_{s-1}(x_i)$ en utilisant la relation de récurrence (5.2).

Exercice 5.2

1. Écrire une fonction pour calculer les poids et les abscisses (ou points) d'intégration On pourra comparer les poids et les abscisses d'intégration pour $s = 8$ avec les valeurs présentées dans le tableau ci-dessous

x_i	w_i
$\pm$0.18343464249565	0.36268378337836
$\pm$0.52553240991633	0.31370664587789
$\pm$0.79666647741363	0.22238103445337
$\pm$0.96028985649754	0.10122853629036

2. Écrire un script pour tester la formule de quadrature sur des intégrales dont on connaît la valeur exacte. Le script appellera d'abord la fonction `xwGauss` puis comparera les valeurs exactes et approchées pour plusieurs fonctions. En particulier, on vérifiera que la formule (5.7) est exacte pour les polynômes de $\mathbb{P}_{2s-1}$ et on comparera les valeurs exactes et approchées de l'intégrale de la fonction e^x.

La solution de cet exercice se trouve en page 120.

5.3 DÉVELOPPEMENT D'UNE FONCTION EN SÉRIE DE LEGENDRE

Pour une fonction $f \in L^2(-1,1)$, on définit la série de Legendre $\mathcal{L}(f) = \sum_{j=0}^{\infty} \hat{f}_j L_j$, où les coefficients $\hat{f}_j$ sont donnés par

$$\hat{f}_j = \frac{2j+1}{2} \int_{-1}^{1} f(x) L_j(x) dx \tag{5.9}$$

On définit une approximation de la fonction f par sa série tronquée à l'ordre p (voir page 62)

$$\mathcal{L}_p(f) = \sum_{j=0}^{p} \hat{f}_j L_j.$$

Cette approximation est exacte pour les polynômes de degré inférieurs à p puisque les $(L_j)_{j=0}^{p}$ forment une base orthogonale de $\mathbb{P}_p$. Le calcul des coefficients $\hat{f}_j$ doit se faire par une formule de quadrature approchée. L'erreur induite par cette approximation des termes $\hat{f}_j$ doit être du même ordre ou négligeable devant l'erreur globale de la méthode, il faut donc utiliser une quadrature d'ordre élevée, et pour cela la méthode d'intégration de Gauss-Legendre étudiée au paragraphe précédent est toute indiquée.

Exercice 5.3

1. Écrire un script qui calcule la série de Legendre $\mathcal{L}_p(f)$ d'une fonction f à l'ordre p, comprenant les étapes suivantes
   - Calcul des abscisses et poids d'intégration pour la formule (5.7) pour un s fixé.
   - Calcul approché des coefficients $(\hat{f}_k)_{k=0}^{p}$ définis par (5.9).
   - Représentation sur le même graphique de la fonction f et de sa série $\mathcal{L}_p(f)$ sur l'intervalle $[-1,1]$.
2. Comparer avec l'exemple représenté sur la figure 5.2 pour :
$$f(x) = \sin(6x)\exp(-x).$$
3. Discuter le choix du nombre de points d'intégration s en fonction du degré p de la série de Legendre.

4. Exécuter le script avec des fonctions moins régulières. (On pourra utiliser par exemple les fonctions `abs(x)` et `sign(x)`.)

5. Tracer la courbe d'erreur en norme infinie entre la fonction et sa série en fonction du degré de la série.

La solution de cet exercice se trouve en page 121.

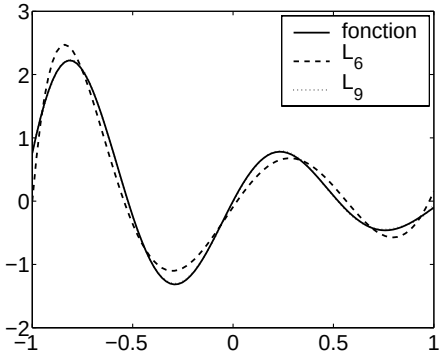


Figure 5.2 Approximations de la fonction $f(x) = \sin(6x)\exp(-x)$ avec $\mathcal{L}_6(f)$ et $\mathcal{L}_9(f)$.

Le choix du nombre de points d'intégration s dépend du degré de la série qu'on veut calculer. En effet, on sait que la formule de quadrature (5.7) sur s points est exacte pour les polynômes de $\mathbb{P}_{2s-1}$. Par ailleurs, la décomposition en série de Legendre tronquée à l'ordre p d'un polynôme de degré p est en fait une décomposition dans la base des $(L_i)_{0\leqslant i\leqslant p}$. On aimerait donc que le calcul des coefficients $\hat{f}_k$ soit exact si f est un polynôme de degré p. Dans ce cas le terme de plus haut degré, $\hat{f}_p$ consiste à intégrer un polynôme de degré $2p$, il faut donc que s soit au moins égal à $p+1$ pour assurer l'exactitude de la formule de quadrature. On laisse au lecteur le soin de vérifier numériquement que si la fonction est de classe C^∞, l'erreur en norme L^∞ décroît plus vite que n'importe quel polynôme en fonction de l'inverse du degré de la série, comme sur la figure 5.3. (Sur cette figure l'erreur ne décroît plus à partir de $p \approx 20$ car elle a alors atteint le seuil de précision de la machine.)

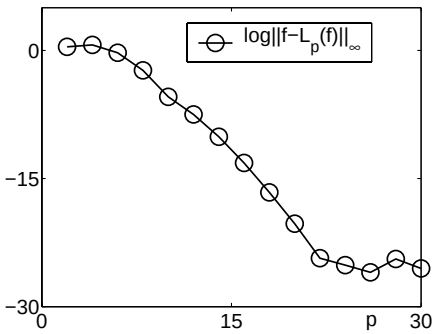


Figure 5.3 Erreur en norme L^∞ entre $f(x) = \sin(6x)\exp(-x)$ et sa série tronquée à l'ordre p, en fonction de p.

Quand on fait tourner le script pour une fonction moins régulière, c'est à dire qui n'est pas de classe C^∞, les résultats se détériorent : pour la fonction $|x|$, l'approximation converge lentement quand on augmente le degré de la série et il reste toujours une erreur visible à l'œil nu près de la singularité. La figure 5.4 montre la dépendance linéaire de l'erreur en fonction de $1/p$.

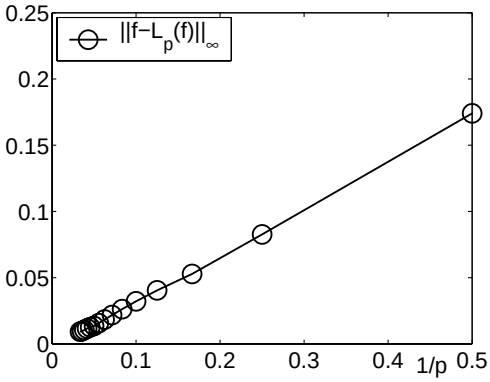


Figure 5.4 Erreur en norme L^∞ entre $f(x) = |x|$ et sa série tronquée à l'ordre p, en fonction de $1/p$.

Pour la fonction discontinue `sign`, la série ne converge pas en norme L^∞. Elle présente des oscillations d'amplitude bornée mais dont la fréquence augmente avec le degré, c'est le phénomène de Gibbs. La figure 5.5 représente les fonctions $|x|$ et sign(x) et leur séries tronquées à l'ordre 30.

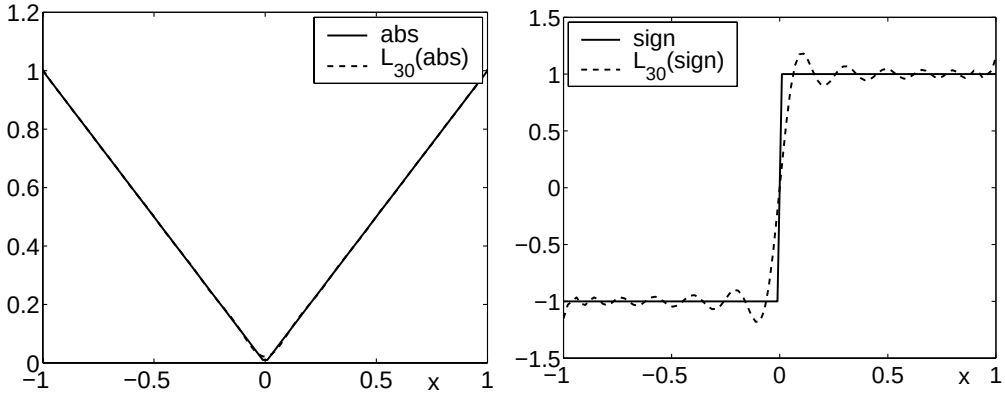


Figure 5.5 Comparaison des fonctions $|x|$ (à gauche) et sign(x) (à droite) avec leurs séries tronquées à l'ordre 30.

5.4 RÉSOLUTION D'UNE ÉQUATION DIFFÉRENTIELLE PAR MÉTHODE SPECTRALE

On rappelle la *formulation variationnelle* du problème (5.1).

Une solution régulière u (*i.e. dans* $H_0^2(-1,1)$) du problème aux limites (5.1) est aussi solution du problème suivant

$$\begin{cases} \text{Trouver } u \in H_0^1(-1,1) \text{ tel que, pour tout } v \in H_0^1(-1,1), \text{ on ait} \\ \displaystyle\int_{-1}^{1} u'(x)v'(x)dx + c\int_{-1}^{1} u(x)v(x)dx = \int_{-1}^{1} f(x)v(x)dx. \end{cases} \tag{5.10}$$

Et réciproquement : toute solution régulière de (5.10) est une solution de (5.1). Le problème (5.10) est appelé « formulation variationnelle » du problème (5.1).

Résolution par méthode spectrale du problème (5.10)

On considère le sous-espace vectoriel de $\mathbb{P}_m$

$$\mathbb{P}_m^0(\Omega) = \{p \in \mathbb{P}_m, p(-1) = p(1) = 0\}.$$

Cet espace est de dimension $m-1$ et s'écrit aussi sous la forme

$$\mathbb{P}_m^0(\Omega) = \{p = (1-x^2)q, \quad q \in \mathbb{P}_{m-2}\}.$$

Il est clair que $\mathbb{P}_m^0(\Omega) \subset H_0^1(\Omega)$ et l'on peut donc sans difficulté définir une méthode d'approximation variationnelle, dite méthode spectrale variationnelle de Galerkin.

$$\begin{cases} \text{Trouver } u_m \in \mathbb{P}_m^0(\Omega) \text{ tel que, pour tout } v_m \in \mathbb{P}_m^0(\Omega), \text{ on ait} \\ \displaystyle\int_{-1}^{1} u_m'(x)v_m'(x)dx + c\int_{-1}^{1} u_m(x)v_m(x)dx = \int_{-1}^{1} f(x)v_m(x)dx. \end{cases} \tag{5.11}$$

On choisit comme base de $\mathbb{P}_m^0(\Omega)$ la famille de polynômes $\mathcal{F}_m = (F_i)_{1\leqslant i\leqslant m-1}$

$$F_i = (1-x^2)L_i'$$

où les L_i sont les polynômes de Legendre étudiés précédemment. La forme matricielle de la formulation variationnelle s'obtient en écrivant la fonction inconnue u_m dans cette base

$$u_m = \sum_{i=1}^{m-1} u_{m,i}F_i \tag{5.12}$$

et en faisant varier v_m dans $\mathcal{F}_m$ dans la formulation variationnelle (5.11).

Construction du système linéaire.

En injectant la combinaison linéaire (5.12) dans (5.11) et en choisissant comme fonction test successivement F_i, pour $i = 1, \ldots m-1$ on obtient un système linéaire de $m-1$ équations dont la solution est le vecteur $\overline{u}_m = (u_{m,i})_{1\leqslant i\leqslant m-1}$ des coefficients de u_m dans la base $\mathcal{F}_m$

$$A_m\overline{u}_m = b_m,$$

où

$$(A_m)_{i,j} = \frac{[i(i+1)]^2}{i+1/2}\delta_{i,j} + c\frac{i(i+1)}{(2i+1)}\frac{j(j+1)}{(2j+1)}\left(\frac{4(2i+1)\delta_{i,j}}{(2i-1)(2i+3)} - \frac{2\delta_{i,j-2}}{2i+3} - \frac{2\delta_{i,j+2}}{2i-1}\right),$$

$$(b_m)_i = \int_{-1}^{1} f(x)F_i(x)dx = \frac{2i(i+1)}{2i+1}\left(\frac{1}{2i-1}\hat{f}_{i-1} - \frac{1}{2i+3}\hat{f}_{i+1}\right).$$

Une fois résolu le système linéaire, on connaît les coefficients spectraux de la solution $\overline{u}$ dans la base $\mathcal{F}_m$. On obtient facilement l'expression de $\overline{u}$ dans la base $(L_j)_{j=0}^m$ en utilisant les relations

$$(1-x^2)L_i' = \frac{i(i+1)}{2i+1}(L_{i-1} - L_{i+1}). \tag{5.13}$$

Exercice 5.4

1. Écrire un programme pour tester cette méthode, comprenant les étapes suivantes
   - Construction de la matrice A_m.
   - Construction du second membre b_m. Cette étape comprend le calcul des coefficients $\hat{f}_k$.
   - Résolution du système linéaire.
   - Calcul des coefficients $\hat{u}_{m_k}$ de la solution numérique, en utilisant la relation (5.13).
   - Représentation sur le même graphique de la fonction u et de la solution numérique u_m sur l'intervalle $[-1, 1]$.

2. Construction d'un cas test : choisir une fonction $u(x)$ telle que $u(-1) = u(1) = 0$ et calculer $f(x)$ de telle manière que u soit solution du problème (5.1), avec c constante. Programmer les deux fonctions $u(x)$ et $f(x)$. La première servira pour comparer la solution obtenue par la méthode spectrale avec la solution exacte.

3. Avantages de la méthode : comparer avec la solution obtenue par une discrétisation par différences finies conduisant à la résolution d'un système linéaire de taille identique (voir paragraphe 8.2). Obtient-on la même précision ? Calculer l'erreur en norme infinie entre la solution exacte et la solution spectrale. Augmenter le nombre de points de discrétisation jusqu'à l'obtention de la même précision par différences finies. Conclure.

La solution de cet exercice se trouve en page 122.

5.5 EXTENSIONS POSSIBLES

Le paragraphe sur les méthodes de quadrature peut donner lieu à plusieurs extensions. On peut en effet utiliser une méthode analogue pour calculer les abscisses et les poids correspondant à d'autres formules de quadrature utilisant d'autres familles

de polynômes orthogonaux. Par exemple, pour le calcul des intégrales sur $\mathbb{R}$ avec la formule approchée dite de Gauss-Hermite :

$$\int_{-\infty}^{+\infty} e^{-x^2} f(x)dx = \sum_{i=1}^{n} \omega_i f(x_i) + R_n \tag{5.14}$$

les x_i sont les zéros des polynômes de Hermite et les poids ω_i sont donnés par la formule

$$\omega_i = \frac{2^{n-1} n! \sqrt{\pi}}{(nH_{n-1}(x_i))^2}.$$

Le reste est

$$R_n = \frac{n!\sqrt{\pi}}{2^n (2n)!} f^{(2n)}(\xi).$$

On utilisera cette fois les polynômes de Hermite définis par

$$\begin{cases} H_0(x) = 1, \\ H_1(x) = 2x, \\ 2xH_n(x) = H_{n+1}(x) + 2nH_{n-1}(x). \end{cases}$$

En ce qui concerne l'application des méthodes spectrales à la résolution des EDPs. On peut penser à généraliser l'exemple étudié dans ce projet dans une dimension plus élevée. En dimension 2 ou 3, les méthodes spectrales restent caractérisées par l'approximation par des polynômes de haut degré et cette fois-ci par l'utilisation de bases tensorisées de polynômes. Elles sont de haute précision et s'avèrent très efficaces dans des géométries simples, parallélépipèdes ou cylindres par exemple. La discrétisation des équations de Laplace dans un carré ou un cube est entièrement détaillée, dans l'ouvrage très récent de [Bernardi, Maday et Rapetti, 2004], avec plusieurs types de conditions aux limites possibles (Dirichlet, Neumann et mixtes). Des problèmes détaillés sont aussi proposés à la fin de cet ouvrage, pouvant constituer des extensions du cas traité ici : la discrétisation spectrale du problème de Dirichlet dans un domaine axisymétrique et la discrétisation spectrale de l'équation de la chaleur en une dimension d'espace.

5.6 SOLUTIONS ET PROGRAMMES

Solution de l'exercice 5.1

Le calcul de la combinaison linéaire (5.5) est fait dans la fonction `CombLinLeg.m`. Pour évaluer dans un tableau `pol` les valeurs du polynômes de Legendre de degré p aux points $x_1, \ldots, x_n$, ce n'est pas la peine de stocker les valeurs de tous les polynômes de degré inférieur à p : seuls les degrés $p-1$ et $p-2$ intervenant dans la relation de récurrence (5.2) doivent être conservés, dans deux tableaux `pol1` et `pol2`. Les valeurs de la combinaison linéaire sont stockées dans un tableau `y` auquel on ajoute les termes $c_i p_i(x)$ au fur et à mesure de leur calcul.

La représentation graphique de $L_0 - 2L_1 + 3L_5$ sur l'intervalle $[-1, 1]$ est faite dans le script `PlotPolLeg.m`. On passe en argument à la fonction `CombLinLeg` le tableau $[0; -2; 0; 0; 0; 3]$ des coefficients de cette combinaison, ainsi que le tableau des points $x_i = -1+(i-1)/250$ pour $i = 1, \ldots, 501$ des points où on veut représenter cette fonction. On obtient le graphe de la figure 5.1.

Il existe une fonction MATLAB , de syntaxe d'appel `L=legendre(n,x)` qui renvoie un tableau L de $n + 1$ lignes, dont la $m + 1$-ième ligne contient les valeurs de la fonction de Legendre L_n^m définie par

$$L_n^m(x) = (-1)^m(1 - x^2)^{m/2}\frac{d^m}{dx^m}L_n(x), \tag{5.15}$$

aux points spécifiés dans le vecteur x. Donc on peut calculer les valeurs du polynôme de Legendre avec cette fonction, en n'utilisant que la première ligne du tableau renvoyé en sortie. Une autre méthode d'évaluation de la combinaison linéaire consiste alors à appeler la fonction `legendre` pour tous les degrés de 0 à p, à extraire la première ligne de l'argument renvoyé et à le multiplier par le coefficient de la combinaison linéaire correspondant.

Le script `PlotPolLeg` compare aussi les temps de calcul des deux méthodes, en appellant la fonction `tic` avant l'appel de `CombLinLeg` et la fonction `toc` juste après. La valeur renvoyée par `toc` contient le temps d'exécution. On refait la même chose avant et après le groupe de commandes pour la méthode utilisant `legendre`. Pour que les temps de calculs soient significatifs, il vaut mieux augmenter le nombre de points de calcul à 500 points sur l'intervalle $[-1, 1]$ ainsi que le degré de la combinaison linéaire à 50. Le rapport entre les temps de calcul, alors supérieur à 100, est incontestablement en faveur du script `CombLinLeg.m`, l'utilisation de la fonction `legendre` dans ce contexte impliquant une grande quantité de calculs inutiles ou bien redondants.

Solution de l'exercice 5.2

Le calcul des abscisses et poids d'intégration de Gauss se fait dans la fonction `xwGauss.m`. Les abscisses sont les valeurs propres de la matrice M, on construit donc cette dernière et on utilise la fonction MATLAB `eig` pour obtenir son spectre.

Une fois les poids et abscisses calculés dans deux vecteurs colonnes `x` et `w`, la formule de quadrature (5.7) se programme à l'aide d'une seule instruction en MATLAB , consistant à faire le produit scalaire du vecteur `w` avec le vecteur des valeurs de la fonction à intégrer aux abscisses `x`.

```
I=w'*f(x);
```

Le script `TestIntGauss.m` teste la formule de quadrature sur une fonction régulière, ici la fonction e^x, et compare également cette méthode d'intégration avec la méthode proposée par MATLAB , programmée dans la fonction `quad`. La syntaxe d'appel est la suivante :

```
q = quad(@fun,a,b)
```

Cette commande renvoie une valeur de l'intégrale de la fonction définie dans `fun.m` entre les bornes `a` et `b` approchée à 10^{-6} près. L'algorithme utilisé est une variante adaptative de la méthode de Simpson. On peut préciser la précision souhaitée en quatrième argument d'entrée :

```
q = quad(@fun,a,b,preci)
```

On peut également récupérer en sortie le nombre d'appels à la fonction définissant l'intégrande :

```
[q,nb] = quad(@fun,a,b,preci)
```

Pour comparer les performances de la méthode d'intégration de `quad` avec la méthode de Gauss du point de vue temps de calcul, on choisit un nombre de points suffisant pour que la quadrature de Gauss donne la valeur de l'intégrale exacte avec six chiffres significatifs : quatre points suffisent pour la fonction e^x par exemple, et on mesure les temps de calculs respectifs des deux méthodes pour évaluer un grand nombre de fois la même intégrale, à l'aide des fonctions `tic` et `toc` comme dans l'exercice précédent. La méthode de Simpson étant moins précise que la méthode de Gauss de degré 4, elle nécessite plus d'évaluations de l'intégrande pour obtenir la même précision et est donc plus lente. Dans la suite du projet, où on doit faire un grand nombre d'évaluations d'intégrales, il est donc avantageux d'utiliser la quadrature de Gauss.

Solution de l'exercice 5.3

La comparaison de la fonction avec sa série pour obtenir les figures 5.2 et 5.5 est faite dans le script MATLAB `AppSerLeg.m`.

Ce script fait appel à la fonction `CalcSerLeg`, qui reçoit en arguments d'entrée

- `s` : le degré de la quadrature de Gauss utilisée pour évaluer les coefficients 5.9,
- `P` : le degré de la série,
- `npt` : le nombre de points de l'intervalle $[-1, 1]$ où on évalue la série,
- `Test` : le nom de la fonction dont on calcule la série, qui doit être définie `inline` ou bien dans un `m-file` avec la déclaration `y=test(x)`.

La fonction renvoie en sortie

- `x` les `npt` abscisses,
- `y` les valeurs de la série aux abcisses `x`,
- `err` l'erreur en norme sup entre la fonction et sa série, évaluée sur les valeurs aux abscisses `x`.

Cette fonction est aussi utilisée dans le script `BoucleSerLeg.m` pour répondre à la question 5 de l'exercice, illustrée par les figures 5.3 et 5.4. Pour différents degrés, ici compris entre 2 et 30 par pas de 2, on évalue la série de Legendre d'une fonction test et l'erreur en norme sup avec la fonction elle-même. On représente ensuite cette erreur en fonction du degré. Pour une fonction régulière, on s'attend à une convergence

exponentielle, ce que l'on retrouve numériquement pour $f(x) = \sin(6x)\exp(-x)$. Pour des fonctions moins régulières, comme par exemple $f(x) = \text{abs}(x)$, l'erreur diminue proportionnellement à $1/P$. Enfin pour une fonction discontinue, comme $f(x) = \text{signe}(x)$, l'erreur ne tend pas vers 0 quand le degré de la série augmente, en raison du phénomène de Gibbs (voir figure 5.5).

Solution de l'exercice 5.4

On choisit comme cas test la fonction $u(x) = \sin(\pi x)\cos(10x)$ qui vérifie les conditions aux limites $u(-1) = u(1) = 0$, et on la programme dans `speciale.m` En prenant comme second membre

$$f(x) = (\pi^2 + 130)\sin(\pi x)\cos(10x) + 20\pi\cos(\pi x)\sin(10x)$$

et la constante $c = 30$, la fonction u est solution du problème (5.1). Le second membre est programmé dans le M-file `fbe.m` en utilisant la dérivée seconde de la fonction solution, elle-même programmée dans `specsec.m`. Le script ci-dessous permet d'enchaîner les différentes étapes du calcul et finalement de comparer la solution calculée par la méthode spectrale avec la solution par différences finies.

Listing 5.1 *MethSpec.m*

```
m=16;     % degré de l'approximation par Legendre
s=m+1;    % degré de la quadrature de Gauss pour le second membre.
global c
c=30.;
% Construction de la matrice
A=zeros(m,m);
for i=1:m
 A(i,i)=(i*(i+1))^2*(1./(0.5+i)+ 4.*c/((2.*i+1.)*(2*i-1)*(2*i+3)));
end
for i=1:m-2
 A(i,i+2)=-2*c*i*(i+1)*(i+2)*(i+3)/((2*i+1)*(2*i+3)*(2*i+5));
end
for i=3:m
 A(i,i-2)=-2*c*i*(i+1)*(i-2)*(i-1)/((2*i-1)*(2*i-3)*(2*i+1));
end
% Construction du second membre
[absc,poids]=xwGauss(s);
t=fbe(absc); u=t.*poids;
LX0=ones(s,1);
LX1=absc;
C=zeros(m+2,1);
C(1)=t'*poids/2; C(2)=3*u'*LX1/2;
for k=2:m+1
    % calcul de f_k dans c(k+1)
    % calcul des valeurs de L_k aux points d' intégration
    % kL_k=(2k-1)xL_{k-1} -(k-1)L_{k-2}
   LX2=((2*k-1)*absc.*LX1-(k-1)*LX0)/k;
```

```
    C(k+1)=(2*k+1)*u'*LX2/2;
    LX0=LX1;
    LX1=LX2;
end
B=zeros(m,1);
for i=1:m
  B(i)=2*i*(i+1)*(C(i)/(2*i-1)-C(i+2)/(2*i+3))/(2*i+1);
end
% Résolution du système linéaire
U=A\B;
%
% Changement de base
% (1-x^2)L_i'=(i(i+1)/(2i+1)).(L_{i-1} - L_{i+1})
UN=zeros(1,m+2);
for k=1:m
  CC=(k+1)*k*U(k)/(2*k+1);
  UN(k)=UN(k)+CC;
  UN(k+2)=UN(k+2)-CC;
end
%
% Calcul de la solution approchée et de l'erreur
%
n=100;
xa=linspace(-1,1,n);
y=CombLinLeg(xa,UN);
es=norm(y-speciale(xa),inf);
%
% Calcul de la solution différences finies
%
mdf=50; h=2/mdf;
xdf=linspace(-1+h,1-h,mdf-1)';
A=toeplitz([2,-1,zeros(1,mdf-3)])/h^2+c*eye(mdf-1,mdf-1);
B=fbe(xdf); ydf=A\B;
%
% Représentation graphique
%
plot (xa,speciale(xa),xa,y,'-',xdf,ydf,'x')
legend('exacte','spectrale','diff. finies')
fprintf('erreur spectrale= %e diff. finies= %e\n ',...
                              es,norm(ydf-speciale(xdf),inf));
```

Il est clair sur la figure (5.6) obtenue en faisant tourner le script `MethSpec.m` que la méthode spectrale est beaucoup plus précise que la méthode des différences finies puisque la solution approchée dans $\mathbb{P}_{21}^0$ se confond avec la solution exacte. L'erreur en norme sup est 5.10^{-5} alors qu'elle est égale à 6.10^{-2} pour la solution différences finies. Il faut faire le calcul aux différence finies sur environ 800 points pour arriver à une erreur en norme infinie aussi petite qu'avec la méthode spectrale. En revanche, dès que la solution du problème continu n'est pas C^∞, les performances

de la méthode spectrale en terme de précision chutent et deviennent comparables - voire inférieure, à celles des différences finies. Le lecteur pourra vérifier ce point en calculant le second membre f de l'équation (5.1) de manière à ce que la dérivée seconde de la solution exacte soit constante par morceaux.

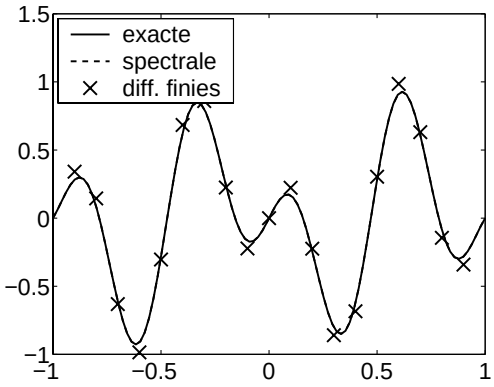


Figure 5.6 Comparaison de la solution exacte, la solution spectrale dans $\mathbb{P}_{21}^0$ et de la solution différences finies avec 20 inconnues.

BIBLIOGRAPHIE

[Delabrière et Postel, 2004] S. DELABRIÈRE, M. POSTEL *Méthodes d'Approximation. Équations Différentielles. Applications Scilab.* Ellipses, Paris (2004).

[Crouzeix et Mignot, 1989] M. CROUZEIX, A.L. MIGNOT *Analyse numérique des équations différentielles.* Masson, Paris, 1984.

[Bernardi, Maday et Rapetti, 2004] C. BERNARDI, Y. MADAY, F. RAPETTI *Discrétisations variationnelles de problèmes aux limites elliptiques* Mathématiques & Applications, Vol. 45, Springer-Verlag, Mai 2004

Projet **6**

Traitement du signal : analyse multiéchelle

Fiche du projet

Difficulté :	2
Notions développées :	Ondelettes, ondelettes de Daubechies, ondelettes de Haar, ondelettes de Schauder, approximation de fonctions, analyse multiéchelle
Domaines d'application :	Traitement du signal, traitement de l'image

On fait dans ce chapitre une rapide introduction à l'analyse multiéchelle (ou multirésolution). Il s'agit d'un domaine des mathématiques très riche en développements théoriques et pratiques. Son application au traitement du signal et de l'image, à l'aide des ondelettes, lui a assuré un succès immédiat. Le fichier des empreintes digitales du FBI, ainsi que le nouveau standard de l'image MPEG3, utilisent la théorie de l'analyse multiéchelle.

6.1 APPROXIMATION D'UNE FONCTION : ASPECT THÉORIQUE

6.1.1 Les fonctions constantes par morceaux

Etudions un exemple simple destiné à mettre en place les différents éléments de l'analyse multiéchelle. Soit f une fonction définie sur l'intervalle $\Omega = [0, 1[$, assez régulière (au sens $f \in L^1(\Omega)$). Pour tout entier naturel $j \geqslant 0$ fixé, on définit les intervalles $\Omega_j^k = [2^{-j}k, 2^{-j}(k+1)[$ pour $k = 0, 1, \ldots, 2^j - 1$, et on considère une approximation $P_j f$ de f par des fonctions constantes par morceaux (voir la figure 6.2). On définit ensuite les valeurs moyennes de f sur les intervalles Ω_j^k

$$P_j^k f = 2^j \int_{\Omega_j^k} f(t)\, dt, \quad \text{pour } k = 0, 1, \ldots, 2^j - 1,$$

Notons $\phi = \chi_{[0,1[}$ la fonction caractéristique de l'intervalle $\Omega = [0, 1[$ et remarquons de suite que cette fonction vérifie la relation (voir la figure 6.1)

$$\forall x \in \Omega, \quad \phi(x) = \phi(2x - 1) + \phi(2x). \tag{6.1}$$

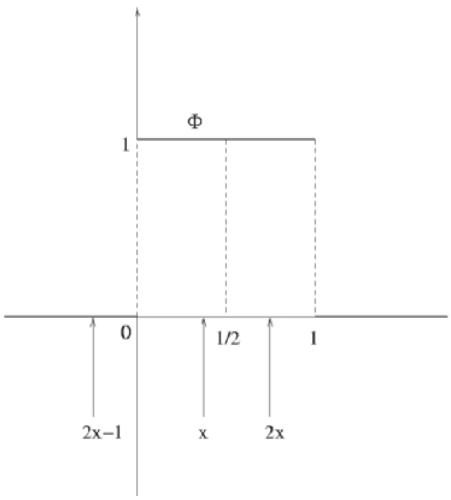


Figure 6.1 La relation d'échelle pour la fonction ϕ.

Dans la théorie de l'analyse multiéchelle, cette relation est appelée *relation d'échelle.* On introduit encore les fonctions χ_j^k, fonctions caractéristiques des intervalles $\Omega_j^k = [2^{-j}k, 2^{-j}(k+1)[$, et on définit la fonction $P_j f$ par

$$\forall x \in \Omega, \quad P_j f(x) = \sum_{k=0}^{2^j-1} P_j^k f \chi_j^k(x) = \sum_{k=0}^{2^j-1} P_j^k f \phi(2^j x - k).$$

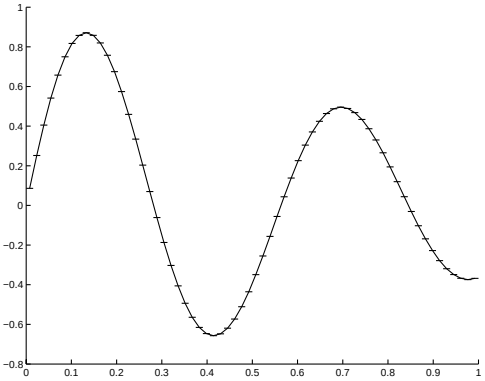


Figure 6.2 Approximation d'une fonction par ses valeurs moyennes ($j = 6$).

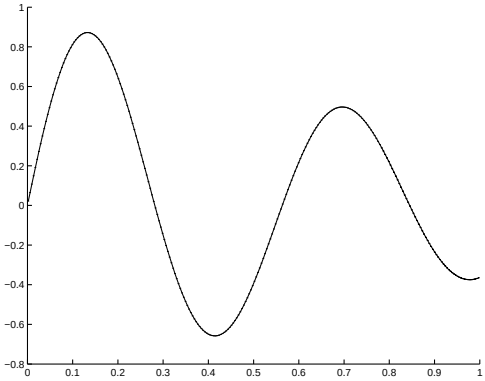


Figure 6.3 Approximation d'une fonction par ses valeurs moyennes ($j = 8$).

Comme le domaine Ω est borné, si on suppose que $f \in L^2(\Omega)$ alors on a $f \in L^1(\Omega)$, et $P_j f$ appartient à l'ensemble V_j défini par

$$V_j = \left\{ f \in L^2(\Omega),\ f_{|\Omega_j^k} = \text{Constante},\ \text{pour } k = 0, 1, \ldots, 2^j - 1 \right\}.$$

Cet espace fonctionnel est de dimension finie : $\dim V_j = 2^j$. On définit ensuite les fonctions ϕ_j^k par

$$\forall x \in \Omega, \quad \phi_j^k(x) = 2^{j/2}\phi(2^j x - k), \qquad \text{pour } k = 0, 1, \ldots, 2^j - 1.$$

On vérifie que les fonctions ϕ_j^k forment une base orthonormale de V_j pour le produit scalaire $L^2 : < f, g >= \int_\Omega f(t)g(t)\, dt$.

Avec ces notations, on peut encore écrire

$$\forall x \in \Omega, \quad P_j(f)(x) = \sum_{k=0}^{2^j-1} < f, \phi_j^k > \phi_j^k(x) = \sum_{k=0}^{2^j-1} c_j^k \phi_j^k(x).$$

Le coefficient

$$c_j^k =< f, \phi_j^k >= \int f(t)\phi_j^k(t)\, dt = 2^{j/2} \int_{\Omega_j^k} f(t)\, dt \tag{6.2}$$

est la *valeur moyenne normalisée* de f sur l'intervalle Ω_j^k, et l'application P_j est la projection orthogonale dans $L^2(\Omega)$ sur V_j. Considérons maintenant deux entiers naturels $j' > j \geqslant 0$, et définissons comme précédemment les espaces $V_{j'}$ et V_j. Les fonctions de base de l'espace $V_{j'}$ sont constantes sur les intervalles $\Omega_{j'}^k$ de taille $2^{-j'}$, tandis que les fonctions de base de l'espace V_j sont constantes sur les intervalles Ω_j^k de taille $2^{-j} > 2^{-j'}$. Comme on a l'inclusion $V_{j'} \subset V_j$, la fonction $P_{j'}f$ constitue une meilleure approximation de f que $P_j f$ au sens où $\|P_{j'}f - f\|_2 < \|P_j f - f\|_2$. On peut montrer que l'approximation $P_j f$ converge vers f dans $L^2(\Omega)$ quand j tend vers l'infini (voir les figures 6.2 et 6.3).

Si $f \in C^0(\Omega)$, alors on peut montrer que l'approximation $P_j f$ converge vers f pour la norme uniforme :

$$\text{si } f \in C^0(\Omega), \quad \text{alors} \quad \lim_{j \to +\infty} \|f - P_j f\|_\infty = 0.$$

Pour $j \geqslant 0$ entier fixé, considérons encore les deux espaces V_j et V_{j+1}. Pour tout $f \in L^2(\Omega)$ et pour $k = 0, 1, \ldots, 2^j - 1$, on écrit en utilisant la relation (6.1)

$$\sqrt{2} \int f(t)\phi_j^k(t)\, dt = \int f(t)\phi_{j+1}^{2k}(t)\, dt + \int f(t)\phi_{j+1}^{2k+1}(t)\, dt$$

ce qui conduit aux relations

$$c_j^k = (c_{j+1}^{2k} + c_{j+1}^{2k+1})/\sqrt{2}, \quad \text{pour } k = 0, 1, \ldots, 2^j - 1. \tag{6.3}$$

Remarque 6.1 : Dans le cas où le domaine (Ω) est $\mathbb{R}$ tout entier, on définit encore l'ensemble

$$V_j = \left\{ f \in L^2(\mathbb{R}),\ f_{|\Omega_j^k} = \text{constante},\ \text{pour } k = 0, 1, \ldots, 2^j - 1 \right\}.$$

avec $\Omega_j^k = [2^{-j}k, 2^{-j}(k+1)]$, et $j \in \mathbb{Z}$.

6.1.2 Décomposition de l'espace V_J

Soit $J \geqslant 0$ un entier fixé. Pour tout entier j tel que $J > j \geqslant 0$, on définit les espaces V_j, $V_{j+1}, \ldots, V_J$, qui vérifient les inclusions $V_j \subset V_{j+1} \subset \ldots \subset V_J$. Pour toute fonction $f \in L^2(\Omega)$, il est naturel de décomposer la projection orthogonale $P_{j+1}f$ de f sur l'espace V_{j+1} comme la somme de la projection de f sur V_j et d'un terme correcteur :

$$P_{j+1}f = P_j f + (P_{j+1}f - P_j f) = P_j f + Q_j f.$$

L'opérateur $Q_j = P_{j+1} - P_j$ ainsi défini est la projection orthogonale dans $L^2(\Omega)$ sur l'ensemble W_j, complémentaire orthogonal de V_j dans V_{j+1} : $V_{j+1} = V_j \oplus W_j$. Introduisons encore la fonction $\psi = \chi_{[0,1/2[} - \chi_{[1/2,1[}$, obtenue comme différence des fonctions caractéristiques des intervalles $[0, 1/2[$ et $[1/2, 1[$. Elle vérifie aussi une relation d'échelle

$$\psi(x) = \psi(2x - 1) - \psi(2x) \tag{6.4}$$

On définit les fonctions ψ_j^k :

$$\forall x \in \Omega, \quad \psi_j^k(x) = 2^{j/2}\psi(2^j x - k), \quad \text{pour } k = 0, 1, \ldots, 2^j - 1.$$

Alors pour toute fonction $f \in L^2(\Omega)$, on calcule les coefficients d_j^k

$$d_j^k = < f, \psi_j^k > = \int f(t)\psi_j^k(t)\, dt. = 2^{j/2} \int_{\Omega_{j+1}^{2k}} f(t)\, dt - 2^{j/2} \int_{\Omega_j^{2k+1}} f(t)\, dt. \tag{6.5}$$

Le coefficient d_j^k est appelé *fluctuation* de f sur l'intervalle Ω_j^k. On écrit alors (voir (6.3))

$$\sqrt{2}\int f(t)\psi_j^k(t)\,dt = \int f(t)\psi_{j+1}^{2k}(t)\,dt + \int f(t)\psi_{j+1}^{2k+1}(t)\,dt.$$

Ce qui conduit aux relations

$$d_j^k = (d_{j+1}^{2k} + d_{j+1}^{2k+1})/\sqrt{2}, \quad \text{pour } k = 0, 1, \ldots, 2^j - 1. \tag{6.6}$$

Par définition des fonctions ϕ et ψ, on vérifie que

$$\forall x \in \Omega, \quad \phi(2^j x - k) + \psi(2^j x - k) = 2\phi(2(2^j x - k)), \quad \text{pour } k = 0, 1, \ldots, 2^j - 1.$$

En reliant cette relation à la définition des fonctions ϕ_j^k et ψ_j^k, on écrit encore

$$2^{-j/2}\phi_j^k + 2^{-j/2}\psi_j^k = 2 \times 2^{-(j+1)/2}\phi_{j+1}^{2k}$$

Ce qui conduit à de nouvelles relations, associées à la décomposition de l'espace $V_{j+1} = V_j \oplus W_j$:

$$c_j^k + d_j^k = \sqrt{2}\, c_{j+1}^{2k}, \quad \text{pour } k = 0, 1, \ldots, 2^j - 1. \tag{6.7}$$

En regroupant les relations (6.3), (6.6) et (6.7) on obtient les relations (6.8) et (6.9), qui sont à la base des algorithmes d'analyse et de synthèse développés dans la suite de cette étude :

$$\begin{cases} c_j^k &= (c_{j+1}^{2k} + c_{j+1}^{2k+1})/\sqrt{2} \\ d_j^k &= (c_{j+1}^{2k} - c_{j+1}^{2k+1})/\sqrt{2}, \quad \text{pour } k = 0, 1, \ldots, 2^j - 1. \end{cases} \tag{6.8}$$

$$\begin{cases} c_{j+1}^{2k} &= (c_j^k + d_j^k)/\sqrt{2} \\ c_{j+1}^{2k+1} &= (c_j^k - d_j^k)/\sqrt{2}, \quad \text{pour } k = 0, 1, \ldots, 2^j - 1. \end{cases} \tag{6.9}$$

Avant d'aller plus loin, constatons qu'il est possible de réitérer le procédé de décomposition de l'espace V_J, pour l'écrire sous la forme

$$V_J = W_{J-1} \oplus V_{J-1} = W_{J-1} \oplus W_{J-2} \oplus V_{J-2} = \cdot s = W_{J-1} \oplus \cdot s W_0 \oplus V_0 \tag{6.10}$$

Sachant que les fonctions ϕ_j^k, pour $k = 0, 1, \ldots, 2^j - 1$, forment une base orthonormale de V_j et que les fonctions ψ_j^k, pour $k = 0, 1, \ldots, 2^j - 1$, forment une base orthonormale de W_j, la relation (6.10) permet de définir $J + 1$ bases orthonormales de V_J. Parmi celles-ci, on distingue la base *canonique* composée des fonctions ϕ_J^k, pour $k = 0, 2, \ldots, 2^J - 1$, dans laquelle on écrit

$$P_J f = \sum_{j=1}^{J-1} \sum_{k=0}^{2^j-1} c_j^k \phi_j^k,$$

et la base de *Haar*, qui utilise toutes les fonctions ϕ_J^k, pour $k = 1, 2, \ldots, 2^J - 1$:

$$P_J f = c_0^0 \phi_0^0 + \sum_{j=1}^{J-1} \sum_{k=0}^{2^j-1} d_j^k \psi_j^k.$$

Rappelons que dans ces formules, $\phi_0^0 = \phi = \chi_{[0,1[}$ est la fonction caractéristique du domaine Ω, et que le coefficient c_0^0 est la valeur moyenne de la fonction f sur Ω : $c_0^0 = \int_\Omega f(t)\,dt$.

> **Remarque 6.2 :** De manière cohérente avec les remarques précédentes, on montre que l'ensemble des fonctions $\{\phi\} \cup \{\psi_j^k\}_{j=0}^{2^J-1}$, base orthonormale de V_J espace de dimension finie, tend vers une base orthonormale de $L^2(\Omega)$ quand J tend vers $+\infty$.

6.1.3 Algorithmes de transformation

Examinons les opérations nécessaires pour passer de la représentation d'une fonction dans la base canonique de V_J à sa représentation dans la base de Haar et réciproquement.

A - Soit f une fonction de $L^2(\Omega)$ et J un entier naturel fixé. On calcule les 2^J coefficients c_J^k, soit de manière exacte si on connait f explicitement, soit de manière approchée en utilisant par exemple un échantillonnage des 2^J valeurs de f aux milieux des intervalles $\Omega_k^J : f(2^{-J}(k+1/2))$. À partir de ces 2^J coefficients c_J^k, on calcule successivement les coefficients $c_j^{k'}$ et $d_j^{k'}$ selon les formules

$$\begin{array}{l}\textbf{pour } j = J-1,\ldots,1,0 \textbf{ faire :}\\ \quad \textbf{pour } k' = 0,1,\ldots,2^j-1 \textbf{ faire :}\\ \qquad c_j^{k'} = (c_{j+1}^{2k'} + c_{j+1}^{2k'+1})/\sqrt{2}\\ \qquad d_j^{k'} = (c_{j+1}^{2k'} - c_{j+1}^{2k'+1})/\sqrt{2}\\ \quad \textbf{fin}\\ \textbf{fin}\end{array} \qquad (6.11)$$

Algorithme d'analyse.

Cette étape est appelée *analyse* ou *décomposition* de la fonction f. À l'étape j, on écrit schématiquement

$$\begin{array}{ccc} & c_j^k & \\ & (0 \leqslant k < 2^j) & \\ \swarrow & & \searrow \\ c_{j-1}^{k'} & & d_{j-1}^{k'} \\ (0 \leqslant k' < 2^{j-1} & & (0 \leqslant k' < 2^{j-1}) \end{array}$$

Une fois calculés, les 2^{j-1} coefficients $d_{j-1}^{k'}$ n'interviennent plus dans les étapes de calcul suivantes : seuls les 2^{j-1} coefficients $c_{j-1}^{k'}$ sont nécessaires au calcul des coefficients $c_{j-2}^{k''}$ et $d_{j-2}^{k''}$. Le coût calcul de l'algorithme (6.11) à l'étape j est donc celui de $2 \times 2^{j-1}$ coefficients, soit $4 \times 2^{j-1}$ opérations. Le coût total de l'algorithme de décomposition de l'espace V_J est alors

$$4 \times \left(2^{J-1} + \ldots + 2^j + \ldots + 2^2 + 1\right) = O(2^J) \text{ opérations.}$$

Remarquez qu'il s'agit d'une valeur optimale au sens où le coût du traitement de $N = 2^J$ données est en $O(N)$ opérations.

B - Dans l'autre sens, si on connait c_0^0, la valeur moyenne de f sur Ω, ainsi que les différents coefficients d_j^k pour $j = 0, \ldots, J-1$, on obtient les coefficients c_j^k par les formules

$$\begin{array}{l} \textbf{pour } j = 0, \ldots, J-1 \textbf{ faire :} \\ \quad \textbf{pour } k' = 0, 1, \ldots, 2^j - 1 \textbf{ faire :} \\ \qquad c_{j+1}^{2k'} = (c_j^{k'} + d_j^{k'})/\sqrt{2}, \\ \qquad c_{j+1}^{2k'+1} = (c_j^{k'} - d_j^{k'})/\sqrt{2}. \\ \quad \textbf{fin} \\ \textbf{fin} \end{array} \tag{6.12}$$

Algorithme de synthèse.

Cette étape est appelée *synthèse* ou *recomposition* de la fonction f. À l'étape j, on écrit schématiquement

$$\begin{array}{ccc} c_{j-1}^{k'} & & d_{j-1}^{k'} \\ (0 \leqslant k' < 2^{j-1}) & & (0 \leqslant k' < 2^{j-1}) \\ & \searrow \quad \swarrow & \\ & c_j^k & \\ & (0 \leqslant k < 2^j) & \end{array}$$

On constate à nouveau, que les 2^{j-1} coefficients c_j^k n'interviennent pas dans les étapes de calcul j' pour $j' > j+1$. Le coût calcul à l'étape j est donc celui de $2 \times 2^{j-1}$ coefficients, soit $4 \times 2^{j-1}$ opérations. Le coût total de l'algorithme de recomposition est encore en $O(N)$ opérations.

La tranformation résumée par les deux algorithmes (6.11) et (6.12) permet de passer de la représentation dans la base canonique à la représentation dans la base de Haar et réciproquement. Il existe d'autres bases orthonormales de l'espace V_J pour lesquelles le calcul des composantes d'un élément de V_J s'obtient à l'aide d'algorithmes semblables à (6.11) et (6.12) . Nous en verrons deux exemples plus loin. Ce type de représentation dans une base orthonormale est appelée *analyse multiéchelle* ou *décomposition multiéchelle.*

6.1.4 Intérêt de la représentation multiéchelle

Examinons maintenant le potentiel de *compression de données* contenu dans la décomposition multiéchelle. Supposons que la fonction f soit constante ($f = C \neq 0$) sur Ω et comparons deux représentations de $P_J f$. Dans la base canonique, chacun des 2^J coefficients c_J^k est égal à C. Pour représenter $P_J f$ dans cette base, un tableau de 2^J coefficients est nécessaire. Pour obtenir l'expression de $P_J f$ dans la base de Haar,

il faut d'abord calculer les coefficients $c_{J-1}^{k'}$ et $d_{J-1}^{k'}$ à partir des c_J^k selon (6.11). On obtient $c_{J-1}^{k'} = C$ et $d_{J-1}^{k'} = 0$ pour $k' = 0, \ldots, 2^{J-1} - 1$. En poursuivant le calcul, on voit que $c_j^k = C$ et $d_j^k = 0$ pour $j = J-1, \ldots, 0$. Finalement dans la base de Haar, il n'y a qu'un seul coefficient non nul : $c_0^0 = C$!

Pour le traitement de l'information, il est très important de pouvoir représenter sous la forme la plus condensée possible tout signal que l'on doit conserver dans une archive ou transmettre sur un réseau. Dans ce but de nombreux algorithmes ont été mis en œuvre pour compresser / décompresser les signaux avec le minimum de perte possible. Sur notre exemple trivial, on comprend intuitivement ce que peut apporter la représentation multiéchelle : pour une fonction f non constante sur Ω, le coefficient c_j^k représente la moyenne de f sur Ω_j^k, tandis que le coefficient d_j^k représente la variation de f à l'échelle j. Si f varie peu sur Ω_j^k, alors d_j^k sera petit et le plus souvent négligeable. Dans ce cas la liste des coefficients d_j^k significatifs dans la base de Haar sera beaucoup plus courte que celle des coefficients c_j^k dans la base canonique. À l'inverse, la présence d'un coefficient d_j^k grand trahit une grande variation de la fonction f dans Ω_j^k. Cette propriété peut être utilisée pour rechercher et localiser de manière automatique les éventuelles singularités d'une fonction f connue sous sa forme $[c_J]$.

> **Remarque 6.3 :** La transformation de Fourier est connue pour permettre la représentation condensée de signaux oscillants. Mais si elle capte très précisément les fréquences présentes dans un signal, elle a le défaut de ne pas fournir une représentation locale en espace, au sens où les fonctions de base utilisées sont des $\cos k\pi x$ ou $\sin k\pi x$, qui oscillent sur Ω tout entier. Ce défaut n'existe pas dans la représentation multiéchelle, puisque les supports des fonctions sont les intervalles Ω_k^j ; cependant, la représentation en fréquences est alors moins précise que celle de Fourier.

6.2 REPRÉSENTATION MULTIÉCHELLE : ASPECT PRATIQUE

Pour prendre la mesure de l'efficacité de la représentation multiéchelle, nous allons traiter un exemple concret. Auparavant, il reste à introduire le mode de stockage des coefficients de la décomposition multiéchelle.

Soit un domaine Ω un intervalle de $\mathbb{R}$, f une fonction définie sur Ω et $J > 0$ un entier fixé. À l'aide de l'algorithme d'analyse (6.8), on calcule pour $j = J-1, \ldots, 0$ les coefficients c_j^k et d_j^k, pour $k = 0, 1, \ldots, 2^j - 1$. Ces coefficients sont stockés au fur et à mesure de leur calcul de la manière suivante : on commence par calculer les coefficients c_J^k suivant la formule (6.2), puis on les range dans un tableau $[c_J]$ de longueur 2^J.

Ensuite on calcule et range les coefficients d_j^k dans un tableau $[d_J]$ de longueur 2^J. On commence à l'étape J par faire $[d_J] = [c_J]$. Puis à l'étape $J-1$ les 2^{J-1} coefficients

$c_{J-1}^{k'}$, calculés selon (6.8), sont rangés dans la première partie du tableau $[d_J]$ (c'est-à-dire les composantes 1 à 2^{J-1}), tandis que les 2^{J-1} coefficients $d_{J-1}^{k'}$, calculés selon (6.8), sont rangés dans la seconde partie du tableau $[d_J]$ (c'est-à-dire les composantes $2^{J-1}+1$ à 2^J). À l'étape $J-2$ les 2^{J-2} coefficients $c_{J-2}^{k''}$ sont à nouveau rangés en tête de tableau (composantes 1 à 2^{J-2}) et "écrasent" ainsi les coefficients $c_{J-1}^{k'}$ pour $k' = 1, 2, \ldots, 2^{J-2}$ devenus inutiles. Les 2^{J-2} coefficients $d_{J-2}^{k''}$ sont rangés à leur suite (composantes $2^{J-2}+1$ à 2^{J-1}) et "écrasent" ainsi les coefficients $c_{J-1}^{k'}$ pour $k' = 2^{J-2}+1, 2, \ldots, 2^{J-1}$. Les coefficients $d_{J-1}^{k'}$ (composantes $2^{J-1}+1$ à 2^J du tableau $[d_J]$ =) ne sont pas modifiés. En procédant ainsi jusqu'à l'étape $J-1$, on obtient finalement le rangement suivant :

$$[d_J] = \left[c_0^0, d_0^0, d_0^1, d_1^1, \ldots, d_0^j, d_1^j, \ldots, d_{2^j-1}^j, \ldots, d_0^{J-1}, d_1^{J-1}, \ldots, d_{2^{J-1}-1}^{J-1}\right]. \quad (6.13)$$

Ce rangement correspond à la décomposition de l'espace V_J suivant le schéma

$$V_J \rightarrow \left\{ \begin{array}{l} W_{J-1} \\ \\ V_{J-1} \end{array} \right. \rightarrow \left\{ \begin{array}{l} W_{J-2} \\ \\ V_{J-2} \end{array} \right. \rightarrow \left\{ \begin{array}{l} \cdots \\ \\ \cdots \end{array} \right. \rightarrow \left\{ \begin{array}{l} W_0 \\ V_0 \end{array} \right.$$

6.3 REPRÉSENTATION MULTIÉCHELLE : MISE EN ŒUVRE

Soit f la fonction définie sur $\Omega = [0, 1[$ par $f(x) = exp(-x)\sin 4\pi x$. On choisit $J = 10$ et on calcule les tableaux $[c_J]$ et $[d_J]$ associés à $P_J f$.

Exercice 6.1

1. Écrire une procédure calculant les coefficients c_j^k suivant (6.2). Ranger ces coefficients dans un tableau $[c_J]$ de longueur 2^J.
2. À l'aide de l'algorithme d'analyse (6.8), calculer pour $j = J-1, \ldots, 0$ les coefficients c_j^k et d_j^k, pour $k = 0, 1, \ldots, 2^j - 1$. Ranger ces coefficients dans un tableau $[d_J]$ de longueur 2^J, suivant la méthode exposée au paragraphe précédent.
3. Écrire une procédure calculant les coefficients c_J^k à partir des éléments du tableau $[d_J]$ selon l'algorithme de synthèse (6.9). Vérifier le résultat.

La solution de cet exercice est présentée à la page 146.

Après avoir programmé les procédures de transformation directe et inverse, on peut effectuer quelques expériences de compression par seuillage.

Exercice 6.2

1. Compter le nombre de coefficients du tableau $[d_J]$ supérieurs à $\varepsilon = 10^{-3}$ en valeur absolue.
2. Recopier le tableau $[d_J]$ dans le tableau $[d_J^\varepsilon]$, en imposant à zéro la valeur des coefficients inférieurs à ε en valeur absolue ($d_j^\varepsilon = 0$ si $|d_j| < \varepsilon$).
3. Construire le tableau $[c_J^\varepsilon]$ à partir du tableau $[d_J^\varepsilon]$ par l'algorithme de synthèse (6.9).
4. Visualiser le résultat obtenu en comparant les courbes représentatives de $P_J f$ et $P_J^\varepsilon f$.
5. Étudier les variations de $\|P_J^\varepsilon f - P_J f\|_2$ et du nombre de coefficients non nuls du tableau $[d_J^\varepsilon]$ en fonction de ε.

La solution de cet exercice est présentée à la page 147.

Remarque 6.4 : D'un point de vue théorique et pratique, il peut être intéressant de choisir un seuil variable en fonction du niveau d'échelle. Par exemple un seuil de la forme $\varepsilon_j = 2^{-j}\varepsilon$, avec ε choisi fixe. Une technique d'élimination des coefficients suivant cette forme de seuil est associée à la norme H^1 plutôt qu'à la norme L^2.

Le tableau 6.1 présente les résultats obtenus pour cette expérience. Il donne pour chaque valeur du seuil le nombre de coefficients tels que $|d_j^k| > \varepsilon$, et l'erreur relative commise sur la donnée initiale, à savoir le rapport $\|P_J^\varepsilon f - P_J f\|_2 / \|P_J f\|_2$. Les figures 6.4 et 6.5 montrent deux signaux reconstitués après seuillage, on remarque la bonne capacité de compression, qui permet de retrouver le signal initial avec une erreur relative de 9% en n'utilisant que 352 coefficients sur 1024.

Tableau 6.1 Compression par seuillage (Haar).

Seuil	Nb. coefficients	Erreur relative
0.1000	78	0.20
0.0500	121	0.16
0.0100	352	0.09
0.0050	563	0.06
0.0010	947	0.02
0.0005	987	0.01
0.0001	1015	0.003

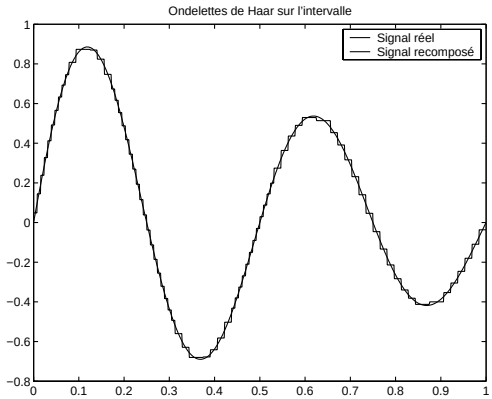


Figure 6.4 Reconstruction après seuillage : $J = 10$, $\varepsilon = 0.10$

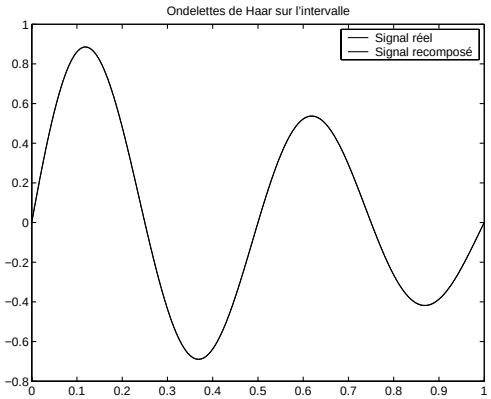


Figure 6.5 Reconstruction après seuillage : $J = 10$, $\varepsilon = 0.01$

6.4 INTRODUCTION À LA THÉORIE DES ONDELETTES

6.4.1 Les fonctions d'échelles et les ondelettes

Si vous avez effectué correctement les expériences numériques du paragraphe précédent, recevez toutes nos félicitations car vous venez de faire vos premiers pas dans le monde des ondelettes ! Quand Haar proposa (en 1910) la construction d'une base orthonormale de $L^2(\Omega)$ sur le modèle précédent, il ignorait l'importance pratique de sa découverte. La théorie des ondelettes n'a vu le jour que dans les années 1960, sous l'impulsion de Morlet, ingénieur dans la prospection pétrolière, et qui cherchait à définir un outil bien adapté au traitement des signaux sismiques, donnant de meilleurs résultats que la transformation de Fourier (voir [Meyer-1990]).

La construction de la base de Haar constitue ainsi le fondement de l'analyse multiéchelle des fonctions de $L^2(\Omega)$. Dans la terminologie actuelle, les fonctions ϕ_j^k sont appelées *fonctions d'échelle*, tandis que les fonctions ψ_j^k sont les *fonctions d'ondelettes*, ou plus simplement les *ondelettes*. À quoi ressemble une ondelette ? Revenons aux expériences numériques précédentes : pour $J = 10$ mettons à zéro le tableau $[d_J]$, puis imposons à la valeur 1 un seul des coefficients d_j^k. L'algorithme de reconstruction (6.9) permet d'obtenir l'ondelette associée ψ_j^k (voir la figure 6.6). Les entiers k et j vérifient $0 < j < J$ et $0 \leqslant k < 2^j$, prenons maintenant un entier $k' \neq k$ tel que $0 \leqslant k < 2^j$ et recommençons l'expérience. Nous obtenons l'ondelette $\psi_j^{k'}$, qui est la translatée de $2^{-j}(k' - k)$ de l'ondelette ψ_j^k (figure 6.6). Ces deux ondelettes appartiennent au sous-espace W_j : ce sont des ondelettes de niveau j. Toutes les ondelettes d'un même niveau (il y en a 2^j au niveau j) se déduisent par translation d'un multiple entier de 2^{-j}. Prenons maintenant un entier j' vérifiant $0 < j' \neq j < J$ et un entier k'' tel que $0 \leqslant k'' < 2^{j'}$. L'ondelette associée $\psi_{j'}^{k''}$ a un air de famille avec les

précédentes : elle a la même forme mais son amplitude et la taille de son support ont été multipliées par $2^{j'-j}$ (figure 6.6).

L'espace $V_J \subset L^2(\Omega)$ est donc la somme directe de de V_0 et de sous-espaces orthogonaux W_j engendrés chacun par 2^j fonctions ondelettes. Quand J tend vers $+\infty$, on retrouve la structure introduite par Haar : l'espace $L^2(\Omega)$ se décompose comme somme de sous-espaces orthogonaux de dimension finie. Vue sous cet angle, la théorie des ondelettes apparait comme un outil puissant pour l'approximation des fonctions. En effet, à chaque niveau on a $V_{j+1} = V_j \oplus W_j$, le sous-espace W_j apparait comme l'ensemble des *fonctions détails* qu'il faut ajouter aux fonctions de V_j pour décrire toutes les fonctions de V_{j+1}. En fixant l'entier $J < +\infty$ on limite la description de l'espace au niveau d'échelle 2^{-J}, c'est-à-dire que l'on ne peut capter la variation $f(x) - f(x')$ d'une fonction avec $|x - x'| < 2^{-J}$; en revanche on peut écrire que $\|P_J f - f\|_2 < C2^{-J}$, pour toute fonction f à variations bornées sur $L^2(\Omega)$. Ainsi on connait à l'avance la précision de l'approximation $P_J f$ d'une fonction f donnée, et le prix à payer pour améliorer ce résultat : il faut calculer 2^J nouveaux coefficients.

Les fonctions d'échelles sont toutes définies par la même fonction ϕ : $\phi_j^k(x) = \phi(2^j x - k)$, tandis que les ondelettes sont définies à partir de la même fonction, la fonction ψ, appelée *ondelette mère* ($\psi_j^k(x) = \psi(2^j x - k)$). Dans ce premier exemple, les deux fonctions ϕ et ψ sont discontinues. Il en résulte que l'approximation $P_J f$ d'une fonction continue f n'est pas continue. Comment obtenir une approximation plus régulière ? De nombreux chercheurs ont travaillé sur cette question. A. Cohen [Cohen-1992] a défini les conditions nécessaires pour qu'un couple de fonctions (ϕ, ψ) génère une analyse multiéchelle d'un espace fonctionnel dans un cadre général, et on sait aujourd'hui construire une approximation $P_J f$ continue d'une fonction continue . La théorie est trop riche pour être détaillée dans ces quelques lignes ; nous nous proposons d'étudier deux exemples d'ondelettes continues : l'*ondelette de Schauder* et une *ondelette de Daubechies.*

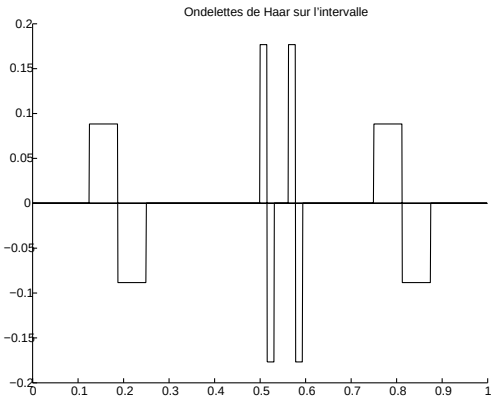


Figure 6.6 Quelques ondelettes de Haar pour $J = 10$.

Remarque 6.5 : L'analyse multiéchelle s'applique aux espaces fonctionnels V de dimension infinie (par exemple $V = L^2(\Omega)$), et conduit à la formule

$$\forall f \in V, \quad f = < f, \phi_0^0 > \phi_0^0 + \sum_{j \geqslant 0} \sum_{k=0}^{2^j - 1} < f, \psi_j^k > \psi_j^k.$$

6.4.2 L'ondelette de Schauder

On reprend la démarche générale en modifiant la définition de l'espace V_J pour $\Omega = [0, 1[$:

$$V_j = \left\{ f \in C^0(\Omega),\ f_{|\Omega_j^k} = \text{Linéaire, pour } k = 0, 1, \ldots, 2^j - 1 \right\} \subset L^2(\Omega).$$

On considère donc l'espace des fonctions linéaires par morceaux et continues sur Ω. Une base de cet espace, bien connue des amateurs d'éléments finis est constituée par les fonctions *chapeaux* φ_j^k (voir la figure 6.7). La fonction φ_j^k est nulle sur $\Omega_j^{k'}$ pour $k' \neq k$. Elle est linéaire sur Ω_{J+1}^{2k} et Ω_{J+1}^{2k+1}, continue sur Ω_J^k et vaut 1 au milieu de Ω_J^k. Soit φ la fonction définie par $\varphi(x) = \max\{0, 1 - |x|\}$. La fonction φ_j^k se déduit de φ par la relation $\varphi_j^k(x) = \varphi(2^j x - k)$. Ceci nous amène à définir les fonctions d'échelle ϕ_j^k, à partir de $\phi = \varphi$, telles que :

$$\forall x \in \Omega, \quad \phi_j^k(x) = 2^{j/2} \phi(2^j x - k), \quad \text{pour } k = 0, 1, \ldots, 2^j - 1.$$

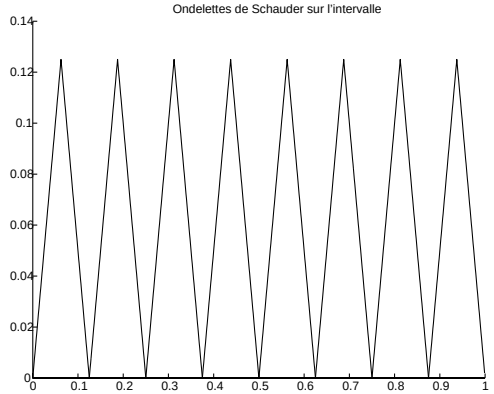


Figure 6.7 Les fonctions φ_J^k pour $J = 3$.

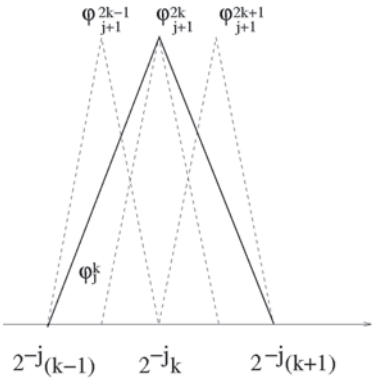


Figure 6.8 La relation d'échelle.

Remarque 6.6 : Pour simplifier l'exposé, nous allons restreindre l'étude aux fonctions définies et périodiques sur Ω intervalle borné . La théorie des ondelettes permet de traiter le cas général, au prix de quelques complications techniques que nous voulons éviter. Remarquez que la fonction chapeau φ_j^0, qui vérifie $\varphi_j^0(0) = \varphi_j^0(1) = 1$ n'est pas présente sur la figure 6.7.

Pour $j \geqslant 0$ entier fixé, considérons encore les deux espaces V_j et V_{j+1}. Pour tout $k = 0, 1, \ldots, 2^j - 1$, on écrit la relation liant les fonctions des niveaux j et $j+1$ (voir la figure 6.8).

$$\varphi_j^k = \varphi_{j+1}^{2k} + \frac{1}{2}\,\varphi_{j+1}^{2k-1} + \frac{1}{2}\varphi_{j+1}^{2k+1}.$$

Pour $j \geqslant 0$ entier fixé, considérons encore les deux espaces V_j et V_{j+1}. Pour tout $f \in C^0(\Omega)$ et pour $k = 0, 1, \ldots, 2^j - 1$, on écrit

$$\sqrt{2}\int f(t)\phi_j^k(t)\,dt = \frac{1}{2}\int f(t)\phi_{j+1}^{2k-1}(t)\,dt + \int f(t)\phi_{j+1}^{2k}(t)\,dt + \frac{1}{2}\int f(t)\phi_{j+1}^{2k+1}(t)\,dt$$

puis on définit les coefficients c_j^k :

$$c_j^k = < f, \phi_j^k > = \int f(t)\phi_j^k(t)\,dt \tag{6.14}$$

ce qui conduit à la relation, analogue à (6.3), et appelée *relation d'échelle* :

$$\sqrt{2}c_j^k = \frac{1}{2}c_{j+1}^{2k-1} + c_{j+1}^{2k} + \frac{1}{2}c_{j+1}^{2k+1} \quad \text{pour } k = 1, 2, \ldots, 2^j - 1. \tag{6.15}$$

Il reste à définir la fonction ψ, mère des ondelettes de Schauder. On admettra que la fonction $\psi = \phi$ convient (ce n'est pas le seul choix possible, mais il a le mérite d'être le plus simple). Cette définition conduit aux nouvelles relations d'analyse

$$\begin{cases} c_j^k &= \sqrt{2}\; c_{j+1}^{2k} \\ d_j^k &= c_{j+1}^{2k+1} - \frac{1}{2}c_{j+1}^{2k+2} - \frac{1}{2}c_{j+1}^{2k}, \quad \text{pour } k = 0, 1, \ldots, 2^j - 1, \end{cases} \tag{6.16}$$

et de synthèse :

$$\begin{cases} c_{j+1}^{2k} &= c_j^k/\sqrt{2} \\ c_{j+1}^{2k+1} &= d_j^k + (c_j^k + c_j^{k+1})/\sqrt{2}, \quad \text{pour } k = 0, 1, \ldots, 2^j - 1, \end{cases} \tag{6.17}$$

qui vont nous permettre d'effectuer de nouvelles expériences numériques. La mise en œuvre pratique de l'analyse multiéchelle associées à ces nouvelles fonctions se résume à utiliser les algorithmes d'analyse et de synthèse en modifiant les relations entre les coefficients. Cette propriété est vraie quelle que soit l'ondelette utilisée ! En effet la structure des calculs du paragraphe précédent, qui permettent de passer de la représentation dans la base canonique à celle dans la base des ondelettes (et inversement), est inchangée. Seuls les coefficients qui interviennent dans les formules sont modifiés (en nombre et valeur). Pour toute forme d'ondelette, le calcul du tableau $[d_J]$ à partir du tableau $[c_J]$ et le calcul inverse selon ce procédé général, est appelé *transformation de Mallat* (voir Mallat [Mallat-1997]).

6.4.3 Mise en œuvre de l'ondelette de Schauder

Reprenons la fonction f définie sur $\Omega = [0, 1[$ par $f(x) = \exp(-x)\sin 4\pi x$, et choisissons encore $J = 10$.

Exercice 6.3

1. À l'aide de l'algorithme d'analyse (6.16), calculer pour $j = J-1, \ldots, 0$ les coefficients c_j^k et d_j^k, pour $k = 0, 1, \ldots, 2^j - 1$. Ranger ces coefficients dans un tableau $[d_J]$ de longueur 2^J, suivant la méthode utilisée au paragraphe 6.2.
2. Écrire une procédure calculant les coefficients c_J^k à partir des éléments du tableau $[d_J]$ selon l'algorithme de synthèse (6.17).
3. Visualiser une ondelette de Schauder et vérifier qu'elle a bien la même forme qu'une fonction d'échelle.

La solution de cet exercice est présentée à la page 147.

Remarque 6.7 : La restriction aux ondelettes périodiques sur Ω se traduit par un traitement spécial des ondelettes non nulles au bord du domaine. Dans les formules (6.16) et (6.17) cela revient à écrire $c_{j+1}^{2j} = c_{j+1}^0$ pour $j = 1, 2, \ldots, J-1$.

Après avoir programmé les procédures de transformation directe et inverse, on effectue à nouveau quelques expériences de compression par seuillage.

Exercice 6.4

1. Compter le nombre de coefficients du tableau $[d_J]$ supérieurs à $\varepsilon = 10^{-3}$ en valeur absolue.
2. Recopier le tableau $[d_J]$ dans le tableau $[d_J^\varepsilon]$, en imposant à zéro la valeur des coefficients inférieurs à ε en valeur absolue ($d_j^\varepsilon = 0$ si $|d_j| < \varepsilon$).
3. Construire le tableau $[c_J^\varepsilon]$ à partir du tableau $[d_J^\varepsilon]$ par l'algorithme de synthèse (6.17).
4. Visualiser le résultat obtenu en comparant les courbes représentatives de $P_J f$ et $P_J^\varepsilon f$.
5. Étudier les variations de $\|P_J^\varepsilon f - P_J f\|_2$ et du nombre de coefficients non nuls du tableau $[d_J^\varepsilon]$ en fonction de ε.

La solution de cet exercice est présentée à la page 149.

Le tableau 6.2 présente les résultats obtenus pour cette expérience. Il donne pour chaque valeur du seuil le nombre de coefficients tels que $|d_j^k| > \varepsilon$, et l'erreur relative commise sur la donnée initiale, à savoir le rapport $\|P_J^\varepsilon f - P_J f\|_2 / \|P_J f\|_2$. Les figures

6.9 et 6.10 montrent deux signaux reconstitués après seuillage, on remarque la très bonne capacité de compression, qui permet de retrouver le signal initial avec une erreur relative de 5.6% en n'utilisant que 66 coefficients sur 1024 !

Tableau 6.2 Compression par seuillage (Schauder).

Seuil	Nb. coefficients	Erreur relative
0.1000	27	0.13
0.0500	36	0.11
0.0100	66	0.056
0.0050	98	0.038
0.0010	187	0.020
0.0005	230	0.016
0.0001	442	0.008

Pour un nombre égal de coefficients du tableau $[d_J]$, on obtient une meilleure approximation de la fonction f avec l'ondelette de Schauder, ce qui est logique puisque l'on a choisi une approximation $P_J f$ continue. Noter que cette amélioration est légèrement plus coûteuse puisque les formules (6.16) et (6.17) comportent plus de coefficients.

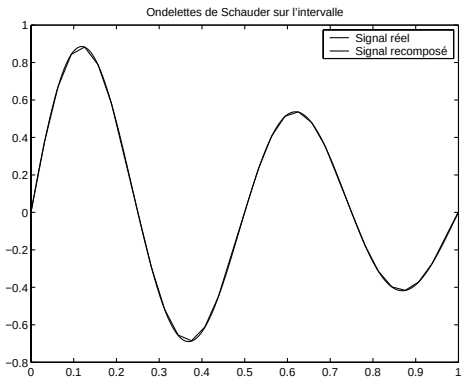


Figure 6.9 Reconstruction après seuillage : $J = 10$, $\varepsilon = 0.10$

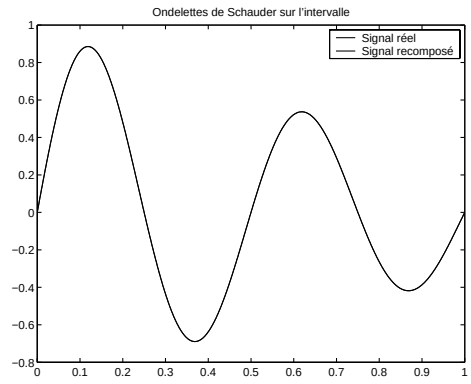


Figure 6.10 Reconstruction après seuillage : $J = 10$, $\varepsilon = 0.01$

6.4.4 L'ondelette 4 de Daubechies

Peut-on faire mieux ? La réponse à cette question est bien connue : oui à condition d'y mettre le prix... A. Cohen [Cohen-1992] a montré que la construction d'une analyse multiéchelle repose sur une relation de la forme

$$\forall x \in \Omega, \quad \phi(x) = \sum_{k\in\mathbb{Z}} h_k \phi(2x-k). \tag{6.18}$$

Cette relation est appelée *relation d'échelle.* Dans la théorie du traitement du signal le tableau h est un *filtre,* et sa définition est suffisante pour construire une analyse multiéchelle, car on peut en déduire une autre relation de la forme

$$\forall x \in \Omega, \quad \psi(x) = \sum_{k\in\mathbb{Z}} \tilde{h}_k \psi(2x-k). \tag{6.19}$$

À l'aide ces deux relations, on peut définir les algorithmes d'analyse et de synthèse associés. C'est bien ce que nous avons fait pour les ondelettes de Haar (6.11) - (6.12) et de Schauder (6.16) - (6.17). I. Daubechies [Daubechies-1992] a montré que la régularité des ondelettes (et donc des fonctions de l'espace V_J) dépend de la taille du filtre, c'est-à-dire du nombre de coefficients h_k non nuls dans (6.18). Elle a ensuite proposé une méthode de construction générale d'ondelettes à support compact et à régularité donnée : les *ondelettes de Daubechies.* Plus il y a de coefficients non nuls dans la relation d'échelle (6.18), meilleure est l'approximation par ondelettes. Pour terminer cette étude on utilise l'ondelette 4 de Daubechies, pour laquelle on donne sans autre justification les formules d'analyse et de synthèse définies à l'aide des coefficients suivants

$$\begin{cases} C_0 = 0,4829629131445341, & C_1 = 0,8365163037378079, \\ C_2 = 0,2241438680420134, & C_3 = -0,1294095225512604. \end{cases} \tag{6.20}$$

Analyse :

$$\begin{cases} c_j^k &= C_0\, c_{j+1}^{2k-1} + C_1\, c_{j+1}^{2k} + C_1\, c_{j+1}^{2k+1} + C_2\, c_{j+1}^{2k+2} \\ d_j^k &= C_3\, c_{j+1}^{2k-1} - C_2\, c_{j+1}^{2k} + C_1\, c_{j+1}^{2k+1} + C_0\, c_{j+1}^{2k+2}, \quad \text{pour } k = 1, 2, \ldots, 2^j - 1. \end{cases} \tag{6.21}$$

Synthèse :

$$\begin{cases} c_{j+1}^{2k} &= C_3\, c_j^{k-1} - C_0\, d_j^{k-1} + C_1\, c_j^k + C_2\, d_j^k \\ c_{j+1}^{2k+1} &= C_2\, c_j^k + C_1\, d_j^k + C_0\, c_j^{k+1} + C_3\, d_j^{k+1}, \quad \text{pour } k = 1, 2, \ldots, 2^j - 1. \end{cases} \tag{6.22}$$

6.4.5 Mise en œuvre de l'ondelette de Daubechies

Reprenons la fonction f définie sur $\Omega = [0, 1[$ par $f(x) = \exp(-x)\sin 4\pi x$, et choisissons encore $J = 10$.

Exercice 6.5

1. À l'aide de l'algorithme d'analyse (6.21), calculer pour $j = J-1, \ldots, 0$ les coefficients c_j^k et d_j^k, pour $k = 0, 1, \ldots, 2^j - 1$. Ranger ces coefficients dans un tableau $[d_J]$ de longueur 2^J, suivant la méthode utilisée au paragraphe 6.2.
2. Écrire une procédure calculant les coefficients c_j^k à partir des éléments du tableau $[d_J]$ selon l'algorithme de synthèse (6.22).
3. Visualiser une ondelette de Daubechies. Quelle forme surprenante ! (voir la figure 6.11)

La solution de cet exercice est présentée à la page 149.

Remarque 6.8 : Comme pour l'exemple précédent, un traitement spécial est réservé aux ondelettes non nulles au bord du domaine.

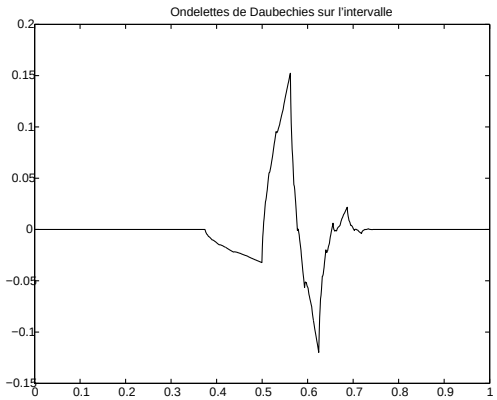


Figure 6.11 Une ondelette de Daubechies.

Après avoir programmé les procédures de transformation directe et inverse, on effectue à nouveau quelques expériences de compression par seuillage.

Exercice 6.6

1. Compter le nombre de coefficients du tableau $[d_J]$ supérieurs à $\varepsilon = 10^{-3}$ en valeur absolue.
2. Recopier le tableau $[d_J]$ dans le tableau $[d_J^\varepsilon]$, en imposant à zéro la valeur des coefficients inférieurs à ε en valeur absolue ($d_j^\varepsilon = 0$ si $|d_j| < \varepsilon$).
3. Construire le tableau $[c_J^\varepsilon]$ à partir du tableau $[d_J^\varepsilon]$ par l'algorithme de synthèse (6.22).
4. Visualiser le résultat obtenu en comparant les courbes représentatives de $P_J f$ et $P_J^\varepsilon f$.

5. Étudier les variations de $\|P_J^\varepsilon f - P_J f\|_2$ et du nombre de coefficients non nuls du tableau $[d_J^\varepsilon]$ en fonction de ε.

La solution de cet exercice est présentée à la page 150.

Le tableau 6.3 présente les résultats obtenus pour cette expérience. Il donne pour chaque valeur du seuil le nombre de coefficients tels que $|d_j^k| > \varepsilon$, et l'erreur relative commise sur la donnée initiale, à savoir le rapport $\|P_J^\varepsilon f - P_J f\|_2 / \|P_J f\|_2$. Les figures 6.12 et 6.13 montrent deux signaux reconstitués après seuillage, on remarque la bonne capacité de compression, qui permet de retrouver le signal initial avec une erreur relative de 5.6% en n'utilisant que 77 coefficients sur 1024.

Tableau 6.3 Compression par seuillage (Daubechies 4).

Seuil	Nb. coefficients	Erreur relative
0.1000	30	0.141
0.0500	43	0.103
0.0100	77	0.056
0.0050	108	0.038
0.0010	207	0.020
0.0005	233	0.017
0.0001	455	0.008

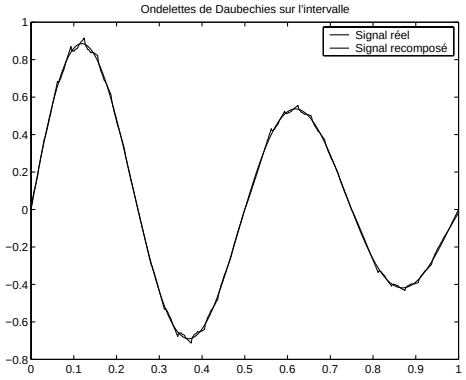


Figure 6.12 Reconstruction après seuillage : $J = 10$, $\varepsilon = 0.10$

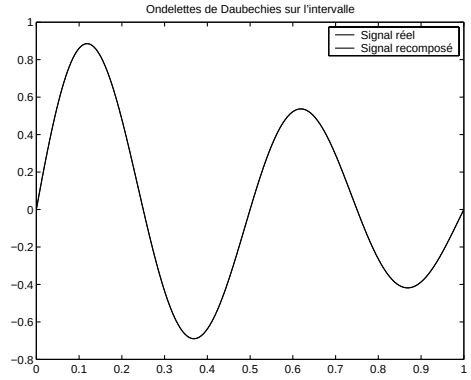


Figure 6.13 Reconstruction après seuillage : $J = 10$, $\varepsilon = 0.01$

À nouveau, pour un nombre égal de coefficients du tableau $[d_J]$, on obtient une meilleure approximation de la fonction f avec l'ondelette de Daubechies.

6.5 GÉNÉRALISATION - TRAITEMENT DE L'IMAGE

Comment étendre les idées développées dans les paragraphes précédents au traitement de l'image ? En construisant des bases d'ondelettes pour l'approximation de fonctions de deux variables. On peut travailler directement à la définition d'une fonction ondelette de deux variables : $\psi_2(x, y)$, cependant la méthode la plus simple consiste à utiliser le produit cartésien, c'est-à-dire prendre $\psi_2(x, y) = \psi(x)\psi(y)$ comme mère des ondelettes. Cela nous conduit à la transformation de Mallat bidimensionnelle suivante : on considére l'image $[c_J]$ à analyser comme une matrice, et on effectue une analyse de chacune de ses lignes pour obtenir le tableau $[\tilde{c}_J]$. On analyse ensuite ce tableau colonne par colonne pour obtenir le tableau $[d_J]$ de la représentation de l'image dans la base des ondelettes. On peut alors effectuer l'opération de seuillage pour compresser ce tableau dans le tableau $[d_J^\varepsilon]$. Pour obtenir le tableau $[c_J^\varepsilon]$, on effectue une synthèse colonne par colonne, suivie d'une synthèse ligne par ligne.

Dans la pratique, on peut effectuer différentes opérations directement à partir de la forme $[d_J]$.

– On peut ainsi comparer deux images distinctes stockées sous cette forme, ce qui permet un examen rapide. C'est le cas par exemple de la comparaison d'empreintes digitales suspectes à celles contenues dans une base de données. Le gain de temps apporté peut être très appréciable. En effet, la comparaison de deux tableaux $[d_J^1]$ et $[d_J^2]$ est beaucoup plus rapide que celle des tableaux $[c_J^1]$ et $[c_J^2]$ correspondants, puisqu'ils ont moins de coefficients.
– On peut aussi utiliser la forme $[d_J]$ pour rechercher une singularité dans une image. En effet, celle-ci sera caractérisée par un coefficient $[d_J]_{k,l}$ grand en valeur absolue. Les astronomes effectuent ainsi une analyse automatique des clichés transmis (sous forme condensée) par les satellites : les objets ponctuels brillants correspondent à des grands coefficients dans le tableau $[d_J]$. Etant donné le nombre "astronomique" de clichés à traiter, cette recherche serait impossible à réaliser sans ce gain de temps substantiel (noter aussi que le transfert de ces multiples photographies envoyées par les satellites n'est rendu possible que par la compression de toute cette information).

Mise en œuvre

Soit F une image définie sous la forme d'un tableau bidimensionnel de pixels $[c_J]$.

Exercice 6.7

1. Écrire les procédures d'analyse et de synthèse de cette image pour les trois ondelettes présentées dans ce chapitre.
2. Effectuer des expériences de compression de l'image pour un seuil ε donné.
3. Comparer les résultats obtenus pour les trois ondelettes.

La solution de cet exercice est présentée à la page 151.

Les figures 6.14 à 6.17 représentent respectivement l'image originale et les images reconstituées après compression du tableau $[d_J]$ avec le seuil $\varepsilon = 10^{-3}$. Le nombre de coefficients du tableau $[d_j^\varepsilon]$ est respectivement $nbc_H^\varepsilon = 2298$, $nbc_S^\varepsilon = 6284$ et $nbc_D^\varepsilon = 1887$ pour une image de 256x256 pixels, soit un tableau $[c_J]$ de 65536 coefficients.

Figure 6.14 Image originale.

Figure 6.15 Image reconstituée (Haar).

Figure 6.16 Image reconstituée (Schauder).

Figure 6.17 Image reconstituée (Daubechies).

L'analyse multiéchelle est très riche en développements théoriques et applications pratiques. De nombreuses recherches sont en cours dans le monde entier, c'est le domaine des mathématiques qui produit aujourd'hui le plus de publications. Pour une étude plus approfondie de cette théorie, la lecture des ouvrages suivants est recommandée : Cohen [Cohen-1992], Daubechies [Daubechies-1992], Mallat [Mallat-1997] et Meyer [Meyer-1990].

6.6 SOLUTIONS ET PROGRAMMES

Toutes les procédures nécessaires à la résolution des exercices de ce chapitre peuvent être téléchargées à partir de la page web du livre.

Solution de l'exercice 6.1

La procédure du fichier `haar.m` réalise les phases d'analyse et de synthèse selon les formules (6.8) et (6.9). L'analyse ($[c_J] \longrightarrow [d_J]$) est effectuée pour la valeur 1 du paramètre *is*, la synthèse ($[d_J] \longrightarrow [c_J]$) pour la valeur -1.

Listing 6.1 *haar.m*

```
       function [ur,uw]=haar(u,uo,nbp,nbni,is);
%
% Transformation en ondelettes de Haar
%
      sqr2=sqrt(2.d0);
      c1=sqr2/2.d0;c2=sqr2/2.d0;
      d1=sqr2/2.d0;d2=-sqr2/2.d0;
%
      if (is==1)
%
% Analyse
%
          uw=u;ur=u;
          nj=nbp;
% calcul des coefficients
          for ni=1:nbni
             for i=1:nj
                ur(i)=uw(i);
             end
             nj=nj/2;
             for i=1:nj
                i2=2*i;
% coefficients sur Vj
                uw(i)=ur(i2-1)*c1+ur(i2)*c2;
% coefficients sur Wj (ondelettes)
                uw(nj+i)=ur(i2-1)*d1+ur(i2)*d2;
             end
          end
%
      else
%
% Synthese
%
          ur=uo;uw=uo;
          nj=1;
```

```
% calcul des coefficients
          for ni=1:nbni
             nj2=nj*2;
             for i=1:nj2
                uw(i)=ur(i);
             end
             for i=1:nj
                i2=2*i;
% coefficients sur Vj+1
                ur(i2-1)=uw(i)*c1+uw(nj+i)*c2;
                ur(i2)=uw(i)*d1+uw(nj+i)*d2;
             end
             nj=nj2;
          end
%
      end
```

La procédure du fichier `imaexo1.m` définit un ensemble de points sur l'intervalle $[0, 1]$ et réalise l'échantillonnage des valeurs de la fonction définie dans le fichier `fo.m` (pour cet exemple particulier $fo(x) = \exp(-x)\sin(4\pi x)$). Cet ensemble de valeurs est ensuite approché par une fonction constante par morceaux à l'aide de `cstemor.m`. La procédure du fichier `haar.m` réalise alors l'analyse et la synthèse d'un signal donné par son échantillonnage sur l'intervalle $[0, 1]$, en utilisant les ondelettes de Haar.

Solution de l'exercice 6.2

La procédure du fichier `imaexo2.m` effectue les mêmes calculs que celle du fichier `imaexo1.m`, à la différence près que tous les coefficients d'ondelettes inférieurs en valeur absolue à un seuil donné sont mis à zéro avant d'effectuer l'étape de synthèse. On réalise ainsi les tests de compression du tableau 6.1 et les figures 6.4 et 6.5.

Solution de l'exercice 6.3

La procédure du fichier `schauder.m` réalise les phases d'analyse et de synthèse selon les formules (6.16) et (6.17). L'analyse ($[c_J] \longrightarrow [d_J]$) est effectuée pour la valeur 1 du paramètre *is*, la synthèse ($[d_J] \longrightarrow [c_J]$) pour la valeur -1.

Listing 6.2 *schauder.m*

```
        function [ur,uw]=schauder(u,uo,nbp,nbni,is);
%
% Transformation en ondelettes de Schauder
%
      sqr2=sqrt(2.d0);
      dsqr2=2.d0*sqr2;
      c1=sqr2;c2=sqr2/2.d0;
```

```
      d1=1.d0;d2=-0.5d0;d3=sqr2/4.d0;
%
      if (is==1)
%
% Analyse
%
          uw=u;ur=u;
          nj=nbp;
% calcul des coefficients
          for ni=1:nbni
             for i=1:nj
                ur(i)=uw(i);
             end
             nj=nj/2;
             for i=1:nj
                i2=2*i;
% coefficients sur Vj
                uw(i)=ur(i2-1)*c1;
% coefficients sur Wj (ondelettes)
                i2p1=i2+1;
                if (i==nj) i2p1=1; end
                uw(nj+i)=ur(i2)+ur(i2p1)*d2+ur(i2-1)*d2;
             end
          end
%
      else
%
% Synthese
%
          ur=uo;uw=uo;
          nj=1;
% calcul des coefficients
          for ni=1:nbni
             nj2=nj*2;
             for i=1:nj2
                uw(i)=ur(i);
             end
             for i=1:nj-1
                i2=2*i;
% coefficients sur Vj+1
                ur(i2-1)=uw(i)*c2;
                ur(i2)=uw(nj+i)+(uw(i)+uw(i+1))*d3;
             end
             ur(2*nj-1)=uw(nj)*c2;
             ur(2*nj)=uw(2*nj)+(uw(nj)+uw(1))*d3;
             nj=nj2;
          end
%
      end
```

La procédure du fichier `imaexo3.m` définit un ensemble de points sur l'intervalle $[0, 1]$ et réalise l'échantillonnage des valeurs de la fonction définie dans le fichier `fo.m`. Cet ensemble de valeurs est ensuite approché par une fonction constante par morceaux à l'aide de `cstemor.m`. La procédure du fichier `schauder.m` réalise alors l'analyse et la synthèse d'un signal donné par son échantillonnage sur l'intervalle $[0, 1]$, en utilisant les ondelettes de Schauder.

Solution de l'exercice 6.4

La procédure du fichier `imaexo4.m` effectue les mêmes calculs que celle du fichier `imaexo3.m`, à la différence près que tous les coefficients d'ondelettes inférieurs en valeur absolue à un seuil donné sont mis à zéro avant d'effectuer l'étape de synthèse. On réalise ainsi les tests de compression du tableau 6.2 et les figures 6.9 et 6.10.

Solution de l'exercice 6.5

La procédure du fichier `daube4.m` réalise les phases d'analyse et de synthèse selon les formules (6.21) et (6.22). L'analyse ($[c_J] \longrightarrow [d_J]$) est effectuée pour la valeur 1 du paramètre *is*, la synthèse ($[d_J] \longrightarrow [c_J]$) pour la valeur -1.

Listing 6.3 *daube4.m*

```
      function [ur,uw]=daube4(u,uo,nbp,nbni,is);
%
% Transformation en ondelettes de Daubechies (daub4)
%
       c0=0.4829629131445341d0;
       c1=0.8365163037378079d0;
       c2=0.2241438680420134d0;
       c3=-0.1294095225512604d0;
%
      if (is==1)
%
% Analyse
%
          uw=u;ur=u;
          nj=nbp;
% calcul des coefficients
          for ni=1:nbni
             for i=1:nj
                ur(i)=uw(i);
             end
             njp1=nj+1;
             if (ni==1) njp1=1; end
             ur(njp1)=ur(1);
             nj2=nj;
             nj=nj/2;
             for i=1:nj-1
                i2=2*i;
```

```
% coefficients sur Vj
                uw(i)=c0*ur(i2-1)+c1*ur(i2)+c2*ur(i2+1)+c3*ur(i2+2);
% coefficients sur Wj (ondelettes)
                uw(nj+i)=c3*ur(i2-1)-c2*ur(i2)+c1*ur(i2+1)-c0*ur(i2+2);
             end
             uw(nj)=c0*ur(nj2-1)+c1*ur(nj2)+c2*ur(1)+c3*ur(2);
             uw(nj2)=c3*ur(nj2-1)-c2*ur(nj2)+c1*ur(1)-c0*ur(2);
          end
%
      else
%
% Synthese
%
          ur=uo;uw=uo;
          nj=1;
% calcul des coefficients
          for ni=1:nbni
             nj2=nj*2;
             for i=1:nj2
                uw(i)=ur(i);
             end
             ur(1)=c2*uw(nj)+c1*uw(nj2)+c0*uw(1)+c3*uw(nj+1);
             ur(2)=c3*uw(nj)-c0*uw(nj2)+c1*uw(1)-c2*uw(nj+1);
             for i=1:nj-1
                i2=2*i;nji=nj+i;
% coefficients sur Vj+1
                ur(i2+1)=c2*uw(i)+c1*uw(nji)+c0*uw(i+1)+c3*uw(nji+1);
                ur(i2+2)=c3*uw(i)-c0*uw(nji)+c1*uw(i+1)-c2*uw(nji+1);
             end
             nj=nj2;
          end
%
      end
```

La procédure du fichier `imaexo5.m` définit un ensemble de points sur l'intervalle $[0, 1]$ et réalise l'échantillonnage des valeurs de la fonction définie dans le fichier `fo.m`. Cet ensemble de valeurs est ensuite approché par une fonction constante par morceaux à l'aide de `cstemor.m`. La procédure du fichier `daube4.m` réalise alors l'analyse et la synthèse d'un signal donné par son échantillonnage sur l'intervalle $[0, 1]$, en utilisant les ondelettes 4 de Daubechies.

Solution de l'exercice 6.6

La procédure du fichier `imaexo6.m` effectue les mêmes calculs que celle du fichier `imaexo5.m`, à la différence près que tous les coefficients d'ondelettes inférieurs en valeur absolue à un seuil donné sont mis à zéro avant d'effectuer l'étape de synthèse. On réalise ainsi les tests de compression du tableau 6.3 et les figures 6.12 et 6.13.

Solution de l'exercice 6.7

Les procédures des fichiers `imaexo7h.m`, `imaexo7s.m` et `imaexo7d.m` lisent le fichier image `lenna.jpg`. Elles effectuent ensuite une analyse, une compression puis une synthèse de cette image initiale en utilisant respectivement les ondelettes de Haar, Schauder et Daubechies. On réalise ainsi les figures 6.15, 6.16 et 6.17. On remarque que l'analyse et la synthèse d'une image par ondelettes sont réalisées par une succession d'analyses et de synthèses de signaux monodimensionnels.

BIBLIOGRAPHIE COMPLÉMENTAIRE OU LECTURE POUR APPROFONDIR

[Cohen-1992] A. COHEN
Ondelettes et traitement numérique du signal,
Masson. Paris (1992)

[Meyer-1990] Y. MEYER
Ondelettes et opérateurs. Tomes I à III,
Hermann. Paris (1990)

[Daubechies-1992] I. DAUBECHIES
Ten lectures on wavelets,
Society for Industrial and Applied Mathematics. Philadelphia. Pennsylvania (1992)

[Mallat-1997] S.G. MALLAT
A wavelet tour of signal processing,
Academic Press. New York (1997)

Projet **7**

Élasticité : déformation d'une membrane

Fiche du projet

Difficulté :	2
Notions développées :	Différences finies 2D, Laplacien, Bi-Laplacien, problème linéaire, problème non-linéaire, problème d'évolution
Domaines d'application :	Mécanique des solides, élasticité linéaire, membrane, plaque

On étudie dans ce projet la déformation de la membrane d'un micro de téléphone soumise aux variations de la pression acoustique engendrées par la voix. Ces variations de pression sont transformées par un condensateur en variations de potentiel, qui à leur tour produisent des variations d'intensité électrique, à l'origine du signal transmis à l'appareil récepteur.

Pour simplifier le traitement de ce problème, on va considérer une membrane de forme rectangulaire et de faible épaisseur, selon le dispositif représenté sur la figure 7.1.

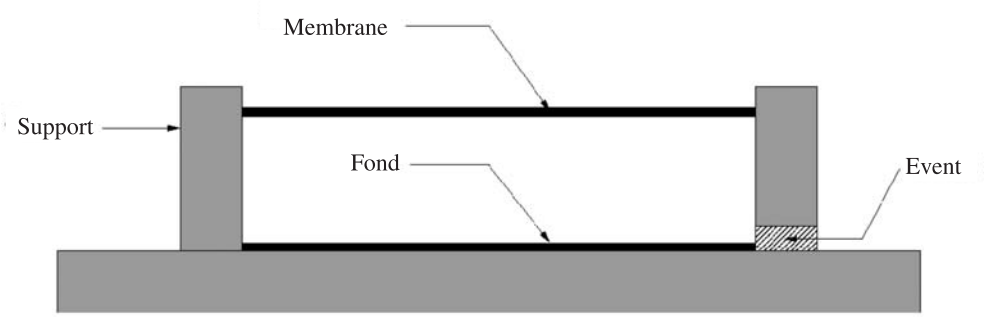


Figure 7.1 Le capteur de pression (vue latérale).

7.1 LE PROBLÈME D'ÉLASTICITÉ (ÉQUATION LINÉAIRE)

La membrane se déforme sous l'effet de la différence de pression entre la cavité et l'extérieur. La cavité est mise à la même pression statique que l'extérieur par un évent. Celui-ci filtre également les variations de pression au-delà de quelques Hertz ; ainsi seules les variations de la pression extérieure (acoustique) déforment la membrane.

Pour une variation connue de la pression acoustique P_a, la déformation de la membrane f_a est solution d'une équation aux dérivées partielles, dont l'opérateur différentiel dépend du modèle choisi. Pour une membrane très tendue, on écrit

$$-c_1 \Delta f_a = P_a. \tag{7.1}$$

tandis que pour une membrane non tendue, on écrit

$$c_2 \Delta^2 f_a = P_a. \tag{7.2}$$

Dans ces équations Δ représente l'opérateur de Laplace en 2D :

$$\Delta f_a = \frac{\partial^2 f_a}{\partial x^2} + \frac{\partial^2 f_a}{\partial y^2} \quad \text{et} \quad \Delta^2 f_a = \Delta(\Delta f_a).$$

Les coefficients c_1 et c_2 sont des constantes associées à des grandeurs physiques que l'on précisera en fonction du matériau composant la membrane. Par souci de généralité, on va traiter le cas d'une membrane « moyennement tendue » :

$$c_2 \Delta^2 f_a - c_1 \Delta f_a = P_a. \tag{7.3}$$

Pour toute variation de pression P_a admissible (c'est-à-dire physique), l'équation (7.3) admet une solution f_a (voir par exemple [Ciarlet-1982]). En fait elle en admet même une infinité puisque $f_a + f_h$ est aussi solution pour toute fonction harmonique f_h. L'unicité de la solution est obtenue en imposant des conditions sur f_a, qui correspondent à l'approche réaliste du problème. Dans notre cas, la membrane est fixée au support tout au long de son bord, ce qui revient à dire que la déformation f_a est nulle au bord.

$$f_{a|\partial\Omega} = 0.$$

Cette condition est suffisante pour garantir l'unicité de la solution de l'équation (7.1) puisque l'opérateur différentiel associé est d'ordre 2 (voir par exemple [Ciarlet-1986]). Pour les équations (7.2) ou (7.3) il faut imposer une condition supplémentaire car l'opérateur différentiel est d'ordre 4. Cette condition s'écrit, en notant $\vec{n}$ la normale extérieure au bord de la membrane :

$$\frac{\partial f_a}{\partial \vec{n}}_{|\partial\Omega} = 0.$$

Elle transcrit mathématiquement la condition d'encastrement de la membrane sur son bord. Notons maintenant

- e : l'épaisseur de la membrane,
- T : la tension mécanique de la membrane,
- E : le module d'Young, constante élastique de la membrane,
- ν : le coefficient de Poisson.

Alors les coefficients c_1 et c_2 sont définis par les relations

$$c_1 = T \quad \text{et} \quad c_2 = \frac{Ee^3}{12(1-\nu)}.$$

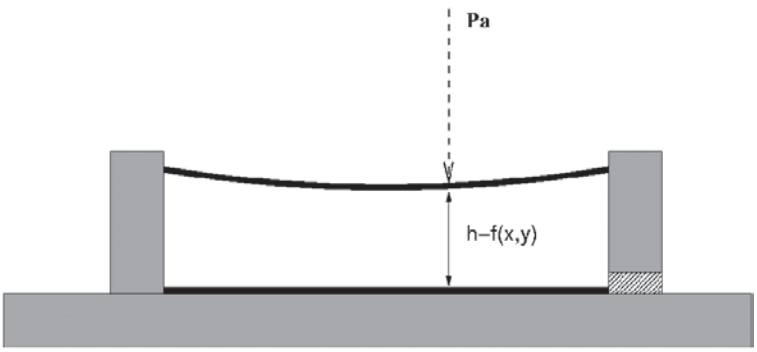


Figure 7.2 La déformation de la membrane.

7.2 LE PROBLÈME ÉLECTROSTATIQUE (ÉQUATION NON-LINÉAIRE)

Oublions un instant les variations de pression acoustique. La membrane et le fond de la cavité sont métallisés. Ils forment les armatures d'un condensateur, dont le diélectrique est l'air. Lorsque ces armatures sont à des potentiels différents, elles sont soumises une force électrostatique qui tend à les rapprocher.

Notons encore

- C : la capacité du condensateur,
- U : la différence de potentiel entre les armatures,
- ε : la permittivité de l'air,
- S : la surface de la membrane (égale à celle du fond de la cavité),
- h : la distance membrane – fond de cavité (entrefer),
- W : l'énergie électrostatique,
- F : la force électrostatique.

En écrivant les relations

$$W = \frac{1}{2}CU^2 = \frac{\varepsilon S U^2}{2h} \quad \text{et} \quad F = -\frac{dW}{dh} = \frac{\varepsilon S U^2}{2h^2}$$

on met en évidence la pression électrostatique :

$$P_e = \frac{F}{S} = \frac{\varepsilon U^2}{2h^2}.$$

Comme au paragraphe précédent, la pression électrostatique déforme la membrane et la déformation f_e associée est solution d'une équation de type (7.3) dont le second membre est P_e. En conséquence la distance membrane – fond de cavité n'est pas constante et on écrit donc en tout point M de coordonnées (x, y) de la membrane

$$P_e(x, y) = \frac{\varepsilon U^2}{2(h - f_e(x, y))^2}.$$

La déformation f_e induite par la pression électrostatique est donc solution d'une équation non-linéaire

$$c_2 \Delta^2 f_e - c_1 \Delta f_e = P_e(f_e). \tag{7.4}$$

L'unicité de la solution f_e est encore assurée par le respect de deux conditions au bord :

$$f_e = 0 \quad \text{et} \quad \frac{\partial f_e}{\partial \vec{n}} = 0. \tag{7.5}$$

7.3 DISCRÉTISATION DU PROBLÈME

D'une manière générale l'expression des données physiques du problème (souvent mesurées expérimentalement) ne permet pas de trouver une solution explicite de l'équation (7.3) avec conditions au bord (7.5). Cette solution ne peut alors être obtenue que par valeurs approchées, à l'aide par exemple de la méthode des *différences finies* (voir par exemple [Euvrard-1990]). Cette méthode consiste à construire un ensemble de points $M_{i,j}$ sur la membrane, disposés selon une grille rectangulaire comprenant *mx* points suivant l'axe des abscisses et *my* suivant l'axe des ordonnées (voir la figure 7.3) puis à calculer les valeurs $f_{i,j} \approx f(M_{i,j})$, approximations aux points $M_{i,j}$ de la grille des valeurs de la solution f de l'équation (7.3) avec conditions au bord. Ces valeurs sont calculées à partir d'une approximation de l'opérateur différentiel.

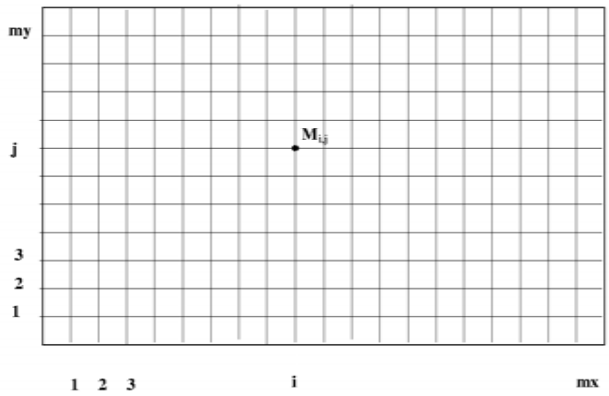


Figure 7.3 La grille à *mx* ∗ *my* points.

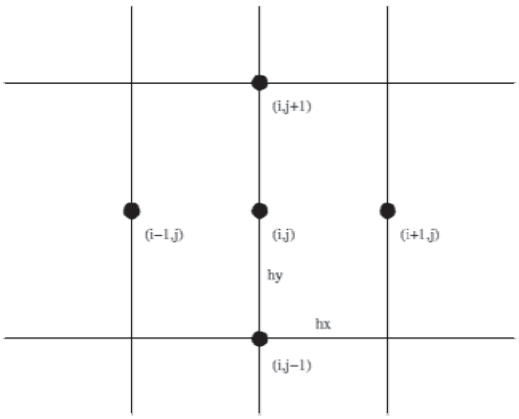


Figure 7.4 Discrétisation du Laplacien. Le schéma à 5 points.

On sait traiter le cas de l'opérateur de Laplace dans le rectangle $[0, L_x] \times [0, L_y]$, avec un pas de maillage en x constant $hx = L_x/(mx + 1)$ et un pas de maillage en y constant $hy = L_y/(my + 1)$. Pour cela, on utilise le schéma à 5 points (voir la figure 7.4) qui s'écrit en tout point $M_{i,j}$ de la grille

$$-(\Delta_5 f)_{i,j} = \frac{1}{hx^2}(-f_{i+1,j} + 2f_{i,j} - f_{i-1,j}) + \frac{1}{hy^2}(-f_{i,j+1} + 2f_{i,j} - f_{i,j-1}) \tag{7.6}$$

Pour l'opérateur du Bi-Laplacien $\Delta^2 = -\Delta(-\Delta)$, on remplace dans (7.6) chaque $f_{i,j}$ par $-(\Delta_5 f)_{i,j}$:

$$\begin{aligned}(\Delta_{13}^2 f)_{i,j} &= \frac{1}{hx^2}(-(\Delta_5 f)_{i+1,j} + 2(\Delta_5 f)_{i,j} - (\Delta_5 f)_{i-1,j}) \\ &\quad + \frac{1}{hy^2}(-(\Delta_5 f)_{i,j+1} + 2(\Delta_5 f)_{i,j} - (\Delta_5 f)_{i,j-1}).\end{aligned}$$

pour obtenir le schéma à 13 points (voir la figure 7.5) :

$$\begin{aligned}(\Delta_{13}^2 f)_{i,j} &= \frac{1}{hy^4} f_{i,j-2} \\ &\quad + \frac{2}{hx^2 hy^2} f_{i-1,j-1} - \left(\frac{4}{hx^2 hy^2} + \frac{4}{hy^4}\right) f_{i,j-1} + \frac{2}{hx^2 hy^2} f_{i+1,j-1} \\ &\quad + \frac{1}{hx^4} f_{i-2,j} - \left(\frac{4}{hx^2 hy^2} + \frac{4}{hx^4}\right) f_{i-1,j} \\ &\quad + \left(\frac{8}{hx^2 hy^2} + \frac{6}{hx^4} + \frac{6}{hy^4}\right) f_{i,j} \\ &\quad - \left(\frac{4}{hx^2 hy^2} + \frac{4}{hx^4}\right) f_{i+1,j} + \frac{1}{hx^4} f_{i+2,j} \\ &\quad + \frac{2}{hx^2 hy^2} f_{i-1,j+1} - \left(\frac{4}{hx^2 hy^2} + \frac{4}{hy^4}\right) f_{i,j+1} + \frac{2}{hx^2 hy^2} f_{i+1,j+1} \\ &\quad + \frac{1}{hy^4} f_{i,j+2}\end{aligned} \tag{7.7}$$

L'équation (7.3) est donc remplacée par les $mx * my$ relations obtenues en écrivant

$$c_2(\Delta_{13}^2 f)_{i,j} - c_1(\Delta_5 f)_{i,j} = P(M_{i,j}) = P_{i,j}$$

en tout point $M_{i,j}$ de la grille pour $i = 1, 2, \ldots, mx$ et $j = 1, 2, \ldots, my$. L'ensemble des relations (7.7) forme un système linéaire dont les inconnues sont les $mx * my$ valeurs $f_{i,j}$, approximations aux points $M_{i,j}$ de la grille des valeurs de la solution f de l'équation (7.3). Il faut donc résoudre ce système linéaire, mais auparavant il reste une dernière étape à effectuer : la prise en compte des conditions aux limites. En effet, comme signalé plus haut, pour une pression P donnée l'équation (7.3) admet une infinité de solutions. Pour obtenir une solution unique, il faut imposer des conditions qui correspondent à la fixation de la membrane sur son support. Ces conditions physiques correspondent à des conditions sur les valeurs $f_{i,j}$, à savoir $f_{i,j} = 0$ en tout point $M_{i,j}$ du bord (condition $f = 0$ au bord) ainsi que les relations $(f_{i,j} - f_{i-1,j})/hx = 0$ et $(f_{i,j} - f_{i,j-1})/hx = 0$ (condition sur la dérivée normale au bord).

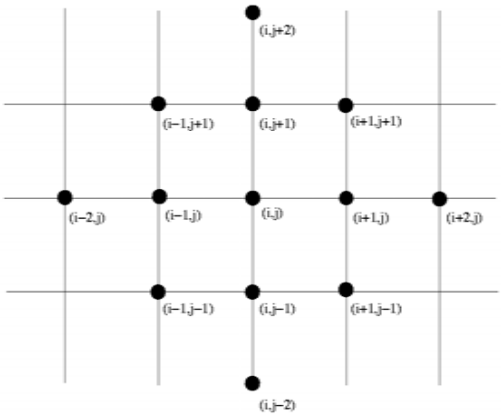


Figure 7.5 Discrétisation du Bi-Laplacien. Le schéma à 13 points.

Remarquer que ces conditions permettent de donner un sens aux formules (7.7) quand $i = 1$ ou $j = 1$. On procède de manière similaire pour $i = mx$ ou $j = my$. Sur le plan pratique il suffit, lors de la construction du système linéaire, de dire que les termes correspondant à des points non existants (par exemple $f_{-1,1}$ présent dans la ligne associée à la valeur $f_{1,1}$) sont nuls.

Implémentation modulaire

Au cours de cette étude de la mise en œuvre de la méthode des différences finies, plusieurs étapes distinctes sont apparues :

1. la construction de la grille des points $M_{i,j}$,
2. l'assemblage du système linéaire,
3. la prise en compte des conditions aux limites,
4. la résolution du système linéaire,
5. la visualisation des résultats.

Dans un code de calcul scientifique utilisant les principes de la programmation structurée, chacune de ces étapes est traitée par un programme spécifique appelé *module* : à une étape déterminée correspond un *module* qui la réalise.

Pour cette étude, nous allons suivre la même démarche et procéder à une mise en place progressive des différents opérateurs numériques nécessaires au calcul. La première étape consiste à négliger l'effet de la pression électrostatique, afin de valider les procédures de résolution du problème linéaire qui seront utilisées ensuite pour la résolution itérative du problème non-linéaire.

7.4 MISE EN ŒUVRE

7.4.1 Notations

Dans la suite, on note

1. $n = mx * my$ le nombre de points de la grille,
2. Ah_5 la matrice associée à l'approximation du Laplacien, par le schéma à 5 points
3. Ah_{13} la matrice associée à l'approximation du Bi-Laplacien par le schéma à 13 points,
4. b le second membre associé à une pression donnée.

7.4.2 Le problème d'élasticité (équation linéaire)

Exercice 7.1

1. Écrire une procédure qui génère les coordonnées (x_i, y_j) des points $M_{i,j}$ de la grille pour $i = 1, 2, \ldots, mx$ et $j = 1, 2, \ldots, my$.
2. Écrire une procédure qui construit la matrice Ah_5 (déclarée creuse) et le second membre b_5 du système linéaire associé à l'équation de Laplace (7.1).
3. Écrire une procédure qui construit la matrice Ah_{13} (déclarée creuse) et le second membre b_{13} du système linéaire associé à l'équation du Bi-Laplacien (7.2).
4. Écrire une procédure qui construit le système linéaire associé à l'équation complète (7.3) avec conditions aux limites.
5. Résoudre ce système linéaire pour une pression donnée.
6. Visualiser les résultats.

Il est conseillé de lire le paragraphe suivant avant de commencer. La solution de cet exercice est présentée à la page 163.

7.4.3 La validation des procédures

On ne peut pas exprimer la solution du problème (7.1) à l'aide de fonctions connues, et c'est pour cela que l'on doit faire une approche numérique. Mais comment sait-on que l'on a calculé la bonne solution ? Est-ce que les procédures sont correctement écrites ? Combien de points sont nécessaires sur la grille pour obtenir une bonne approximation ?

Une façon de répondre à ces interrogations justifiées est d'effectuer la résolution numérique d'un problème dont on connait la solution, puis de comparer le résultat obtenu à celui que l'on doit trouver. Dans le cas de l'équation de Laplace (7.1) on procède de la manière suivante : on choisit une expression plus ou moins compliquée

d'une solution, par exemple celle du fichier `solution.m`

$$\tilde{f}_a(x,y) = 100\sin(3.7\pi x)\sin(5.4\pi y) + (3.7x - 5.4y) \tag{7.8}$$

et on calcule le second membre correspondant, c'est-à-dire (en posant $c_1 = 1$ et $c_2 = 0$ pour simplifier)

$$\tilde{P}_a(x,y) = -\Delta\tilde{f}_a(x,y) = 100(3.7^2 + 5.4^2)\pi^2\sin(3.7\pi x)\sin(5.4\pi y).$$

Cette expression définit le second membre b_5 du point 2 de l'exercice 7.1. La résolution approchée de l'équation de Laplace (7.1) $-\Delta f_a = \tilde{P}_a$, avec la condition aux limites $f_{a|\partial\Omega} = \tilde{f}_{a|\partial\Omega}$, est effectuée en calculant la solution du système linéaire $Ah_5 f_a = b_5$. Cette résolution fournit les valeurs numériques $f_a(x_i, j_j)$, valeurs approchées de la solution de l'équation de Laplace (7.1) aux sommets de la grille. En comparant les $f_a(x_i, j_j)$ aux valeurs exactes $\tilde{f}_a(x_i, j_j)$ on peut apprécier de manière concrète l'erreur associée à cette modélisation, et valider (ou corriger) les procédures de calcul. On procède de même pour la résolution de l'équation du Bi-Laplacien (7.2), puis de l'équation complète (7.3).

7.4.4 Les procédures et l'expérimentation numérique

Les procédures associées à l'exercice 7.1 sont présentées à la page 163. La figure 7.6 montre la solution obtenue après résolution du problème de la membrane. Cette solution numérique est très voisine de la solution exacte (7.8), générée par la procédure du fichier `solution.m` : pour un nombre de points de la grille relativement faible ($nx = 20$ et $ny = 30$), on a déjà une erreur relative

$$\text{Erreur} = \frac{\sum_{i,j} |\tilde{f}_a(x_i, j_j) - f_a(x_i, j_j)|^{1/2}}{\sum_{i,j} |\tilde{f}_a(x_i, j_j)|^{1/2}} = 0,0375.$$

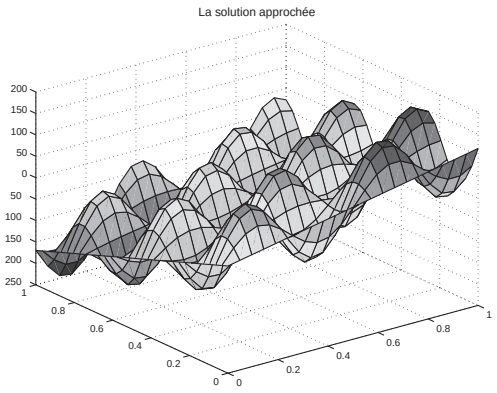


Figure 7.6 La solution du problème linéaire 7.3.

Cette erreur diminue quand on augmente le nombre de points : Erreur = $0,0095$ avec $nx = 40$ et $ny = 60$, mais le temps calcul est alors beaucoup plus long !

7.5 LE PROBLÈME NON-LINÉAIRE

Dans son fonctionnement réel la membrane du microphone est soumise à la fois aux variations de pression acoustique et à la pression électrostatique. Il faut donc résoudre l'équation non-linéaire

$$c_2\Delta^2 f - c_1\Delta f = P_a + P_e(f), \tag{7.9}$$

assortie des conditions au bord. La solution de ce problème est obtenue par un algorithme de point fixe. À partir de la solution du problème linéaire (7.3), que l'on note f_0, on définit la suite $\{f_k\}_{k\in N}$ par l'algorithme de point fixe :

$$c_2\Delta^2 f_{k+1} - c_1\Delta f_{k+1} = P_a + P_e(f). \tag{7.10}$$

La méthode de point fixe est une méthode itérative. Pour ce type de méthode, il faut définir un critère d'arrêt, c'est-à-dire un moyen de décider quand on considère que la solution courante est acceptable. En général on prend un critère basé sur la variation relative de la solution d'une itération à l'autre, de la forme

$$\max_{x,y} |f^{k+1}(x,y) - f^k(x,y)| < \varepsilon \max_{x,y} |f^k(x,y)|.$$

Avant de traiter un problème réaliste, on applique encore le principe de validation des procédures. On choisit donc la solution f, puis on détermine le second membre correspondant. Dans le cas du problème (7.9), cela revient à donner l'expression de la non-linéarité, c'est-à-dire la fonction $P_e\ f :\longrightarrow P_e(f)$. Pour ce calcul de validation, on prendra

$$P_e(f) = \frac{100}{200 - f}.$$

La solution du problème non-linéaire est obtenue après deux itérations de point fixe, pour une valeur $\varepsilon = 0,001$, $nx = 20$ et $ny = 30$. Elle est identique graphiquement à la solution approchée du problème linéaire, représentée sur la figure 7.6. Pour la mise au point des procédures, on pourra s'inspirer de la solution de l'exercice 7.2 à la page 165.

7.6 RÉSOLUTION NUMÉRIQUE DU PROBLÈME NON-LINÉAIRE

Après validation des procédures sur un cas académique, nous pouvons passer à l'étude d'un problème réaliste. Pour cela, il faut affecter des valeurs réelles aux différents paramètres et constantes qui interviennent dans le problème (7.4).

La pression atmosphérique étant égale à 10^5 Pa (Pascal), la pression acoustique P_a varie de 10^{-3} Pa à 10 Pa. On considère que l'oreille humaine perçoit des variations de pression allant de 2.10^{-5} Pa à 2 Pa. Le seuil d'audibilité est donc voisin de 0 dB

(décibel), tandis que le seuil de tolérance se situe vers 100 dB. On pourra prendre, sans dommage pour les oreilles ni pour la procédure, une variation de pression acoustique de l'ordre de 1 Pa.

En ce qui concerne la membrane de silicium, le module d'Young est de $E = 1,3.10^{11}$ Pa (Pascal ou N/m^2, Newton par mètre carré) et le coefficient de Poisson vaut $\nu = 0,25$ (adimensionnel). Le dispositif décrit à la figure 7.1 a comme dimensions typiques : une longueur de 1 mm (millimètre), une largeur de 1 mm et une épaisseur $e = 1$ µm (micron). La tension mécanique de la membrane est $T = 100$ N/m.

Pour le condensateur, l'écartement au repos est $h = 5$ µm. La permittivité de l'air sec est $\varepsilon = 8,85.10^{-12}$ F/m (Faraday par mètre), le potentiel de polarisation de la capacité est $V = 25$ V (Volt). Le maillage de la membrane selon la figure 7.3 comprend $nx = 20$ points dans la direction des abscisses et $ny = 30$ points dans la direction des ordonnées, pour un total de 600 inconnues. La procédure du fichier `pression.m` décrit le comportement non linéaire de la pression pour ce calcul.

Listing 7.1 *pression.m*

```
      function p=pression(x,y,u);
%
% La pression acoustique
      Pa=1.0;   % unité Pascal
% valeurs physiques : varie entre 1.e-03 et 10.
% écartement des armatures du condensateur
      h=5.e-06;   % 5 microns
% surface de la membrane
% S=1.e-09; % 1 mm x 1 mm
% permittivité de l'air
      eps=8.85e-12;   % unité Faraday/m
% potentiel de polarisation de la capacité
      V=25.;   % unité Volt
      hu=h-u;
% La pression électrostatique
      Pe=eps*V*V/(2*hu*hu);
% La pression totale
      p=Pa+Pe;
```

Exercice 7.2

1. Modifier la procédure du fichier `memexo1.m` pour prendre en compte les données du problème linéaire (7.3) dans le cas « réaliste ».
2. Écrire une procédure de mise en œuvre l'algorithme de point fixe
3. Résoudre l'équation non linéaire (7.9).
4. Visualiser les résultats.

La solution de cet exercice est présentée à la page 165. Compte-tenu de toutes les valeurs réalistes, on résout d'abord le problème d'acoustique – équation (7.3) – avec les conditions aux limites (7.5) pour obtenir la déformation représentée à la figure (7.7). La déformation est maximale au centre de la membrane ($\max f_a = 0,080$ μm).

On traite ensuite le problème complet – équation (7.4) – avec les conditions aux limites (7.5). Remarquer qu'il convient de bien calibrer la valeur de la variation de pression acoustique *Pa*, ainsi que celle du potentiel de polarisation *V* pour respecter la contrainte physique $\max f_a < h$. Avec ces données, on obtient la convergence de l'algorithme de point fixe en trois itérations pour le critère $\max |f_e^{k+1} - f_e^k| < 1.\ 10^{-3} \max |f_e^k|$. La déformation (voir la figure (7.8) est maximale au centre de la membrane ($\max f_e = 0,077$ μm).

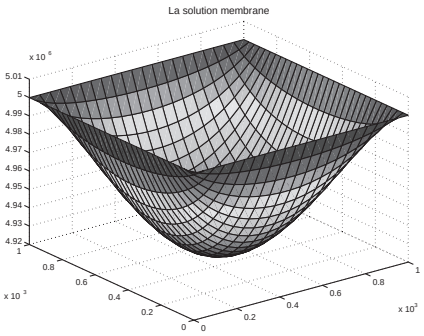


Figure 7.7 La membrane

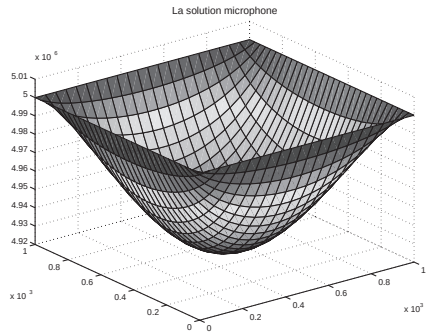


Figure 7.8 Le microphone

7.7 SOLUTIONS ET PROGRAMMES

Toutes les procédures nécessaires à la résolution des exercices de ce chapitre peuvent être téléchargées à partir de la page web du livre.

Solution de l'exercice 7.1

La procédure du fichier `memexo1.m` résout le problème (7.3) pour retrouver la solution connue du fichier `solution.m`. Elle appelle les procédures des fichiers `matrice.m` et `second.m` qui construisent le système linéaire associé à l'équation de Laplace (7.1), puis les procédures des fichiers `matrice2.m` et `second2.m` qui construisent le système linéaire associé à l'équation du Bi-Laplacien (7.2). On obtient ainsi la solution approchée de la figure 7.6.

Listing 7.2 *matrice.m*

```
      function Ah5=matrice(hx,hy,nx,ny);
%%
%% Construction de la matrice du Laplacien 2D : schéma à 5 points
%%
      n=nx*ny;
      h2x=hx*hx;h2y=hy*hy;
      Ah5=sparse(n,n);
      Dx=toeplitz( [2.d0 -1.d0 zeros(1,nx-2) ] );
      Dx = Dx / h2x;
      Dy=eye(nx,nx);
      Dy = - Dy / h2y;
      Dx = Dx - 2.d0 * Dy;
      for k=1:(ny-1)
          i=(k-1)*nx; j=k*nx;
          Ah5( (i+1) : (i+nx) , (i+1) : (i+nx) ) = Dx;
          Ah5( (j+1) : (j+nx) , (i+1) : (i+nx) ) = Dy;
          Ah5( (i+1) : (i+nx) , (j+1) : (j+nx) ) = Dy;
      end;
      i=(ny-1)*nx;
      Ah5( (i+1) : (i+nx) , (i+1) : (i+nx) ) = Dx;
%% figure(10); spy(Ah5); title('La matrice Ah5');
```

Listing 7.3 *matrice2.m*

```
   function Ah13=matrice2(hx,hy,nx,ny);
%%
%% Construction de la matrice du Bi-Laplacien 2D : schéma à 13 points
%%
      n=nx*ny;
      h2x=hx*hx;h2y=hy*hy;
      h4x=h2x*h2x;h4y=h2y*h2y;h4xy=h2x*h2y;
       c1=6.d0/h4x+6.d0/h4y+8.d0/h4xy;
      c2=-4.d0/h4x-4.d0/h4xy;
      c3=1.d0/h4x;
      D=toeplitz( [c1 c2 c3 zeros(1,nx-3) ] );
      c4=-4.d0/h4y-4.d0/h4xy;
      c5=2.d0/h4xy;
      D1=toeplitz( [c4 c5 zeros(1,nx-2) ] );
      c6=1.d0/h4y;
      D2=toeplitz( [c6 zeros(1,nx-1) ] );
      for k=1:(ny-1)
          i=(k-1)*nx; j=k*nx;
          Ah13( (i+1) : (i+nx) , (i+1) : (i+nx) ) = D;
          Ah13( (j+1) : (j+nx) , (i+1) : (i+nx) ) = D1;
          Ah13( (i+1) : (i+nx) , (j+1) : (j+nx) ) = D1;
      end;
```

```
      i=(ny-1)*nx;
      Ah13( (i+1) : (i+nx) , (i+1) : (i+nx) ) = D;
      for k=2:(ny-1)
          i=(k-2)*nx; j=k*nx;
          Ah13( (j+1) : (j+nx) , (i+1) : (i+nx) ) = D2;
          Ah13( (i+1) : (i+nx) , (j+1) : (j+nx) ) = D2;
      end;
%% figure(10); spy(Ah13); title('La matrice Ah13');
```

Solution de l'exercice 7.2

La procédure du fichier `memexo2.m` résout le problème (7.4) pour en trouver la solution physique inconnue. Elle appelle aussi les procédures des fichiers `matrice.m` et `second.m` qui construisent le système linéaire associé à l'équation de Laplace (7.1), puis les procédures des fichiers `matrice2.m` et `second2.m` qui construisent le système linéaire associé à l'équation du Bi-Laplacien (7.2). Les constantes physiques sont prises en compte à leur juste valeur. Enfin la procédure du fichier `pression.m` décrit le terme non-linéaire. On obtient ainsi les solutions des figures 7.7 et 7.8.

BIBLIOGRAPHIE COMPLÉMENTAIRE OU LECTURE POUR APPROFONDIR

[Ciarlet-1982] P. G. Ciarlet *Introduction à l'analyse numérique matricielle et à l'optimisation*, Masson. Paris (1982)

[Ciarlet-1986] P. G. Ciarlet *Elasticité tridimensionnelle*, Masson. Paris (1986)

[Euvrard-1990] D. Euvrard, *Différences finies, éléments finis, méthode des singularités*, Masson. Paris (1990)

Projet **8**

Décomposition de domaines par la méthode de Schwarz

Fiche du projet

Difficulté :	2
Notions développées :	Décomposition de domaines, méthode de Schwarz avec recouvrement, discrétisation du Laplacien, différences finies 1D et 2D
Domaines d'application :	Thermique, régime stationnaire de l'équation de la chaleur

8.1 PRINCIPE ET CHAMPS D'APPLICATION DE LA DÉCOMPOSITION DE DOMAINE

La modélisation réaliste de problèmes physiques passe par des systèmes d'équations aux dérivées partielles, non linéaires dans le cas général, posés sur des domaines qui peuvent être à la fois de forme complexe et de grande taille. Dans la plupart des cas, la méthode numérique sélectionnée nécessite une discrétisation du domaine de la solution, et le nombre de degrés de liberté à déterminer peut rapidement excéder les capacités de calcul de l'équipement informatique. Par exemple, une simulation

d'écoulement autour d'un avion par éléments finis 3D nécessitera une discrétisation du domaine de calcul sur quelques millions de points, avec plusieurs inconnues à déterminer en chaque point... Les méthodes de calcul peuvent en plus être implicites et donc faire intervenir la résolution de systèmes linéaires avec ce nombre impressionnant d'inconnues... Dans le cas où l'on ne dispose pas d'un super calculateur - réservé à quelques centres de recherches très spécialisés - la matrice du système ne tiendra même pas dans la mémoire vive de l'ordinateur !

Une réponse simple à cette difficulté est de découper le problème en plusieurs problèmes plus petits, c'est-à-dire de calculer la solution morceau par morceau comme solution de problèmes posés sur des sous-domaines du domaine initial. Au final, la solution globale sera la juxtaposition des solutions sur chaque sous-domaine. Cette méthode permet aussi de simplifier la résolution de problèmes posés sur des domaines de forme complexe en choisissant un découpage en sous-domaines de forme élémentaire donc plus facile à discrétiser. Une autre extension possible est celle du couplage d'équations pour traiter des interactions entre deux phénomènes physiques différents posés sur des domaines voisins (interaction fluide structure par exemple).

La difficulté posée par cette stratégie est de déterminer les conditions limites à imposer aux bords de chaque sous-domaine. En effet ces bords étant des frontières fictives, la physique du problème ne permet pas d'y fixer des conditions. Deux stratégies, toutes les deux itératives, sont possibles : la première consiste à découper le domaine global en sous-domaines qui se recouvrent partiellement et à utiliser la solution de l'étape précédente sur les sous-domaines voisins pour déterminer les conditions aux limites sur le sous-domaine considéré. La deuxième consiste à découper le domaine sur une partition et à imposer des conditions de continuité aux interfaces.

Une fois la stratégie de découpage adoptée, la méthode de résolution sur chaque sous-domaine sera identique à celle envisagée sur le domaine global, mais fera cette fois-ci intervenir un nombre raisonnable de degrés de liberté. Un exemple simple de résolution par différences finies d'une équation scalaire sur un maillage non structuré découpé en P sous-domaines de même taille N, fera intervenir P résolutions de systèmes linéaires sur chaque domaine, soit un coût en $O(PN^2)$ par itération au lieu de $O((PN)^2)$ pour résoudre une fois le système linéaire permettant d'obtenir la solution sur le domaine entier. Le prix à payer est le caractère itératif de l'algorithme ; la taille et le nombre des sous-domaines doivent être choisis avec soin de manière à rendre l'algorithme le plus compétitif possible. En tout état de cause, même au prix d'une augmentation en temps de calcul, on aura toujours l'avantage - crucial - d'arriver à faire tenir le problème en mémoire sur l'ordinateur dont on dispose. Signalons enfin, même si ce point ne peut évidemment pas être abordé dans le cadre d'un projet MATLAB , que l'atout majeur des méthodes de décomposition de domaines est qu'elles vont de pair avec une implémentation en parallèle. La résolution du problème, identique sous chaque sous-domaine, peut être distribuée sur des processeurs différents, avec des perspectives de gain cette fois ci en temps de calcul aussi bien qu'en place mémoire.

A titre d'exemple, le projet présente une implémentation de la méthode de décomposition de domaine dite méthode de Schwarz avec recouvrement sur le problème modèle du Laplacien 1D et 2D.

$$\begin{cases} -\Delta u(x) + c(x)u(x) = f(x), & \text{pour} \quad x \in \Omega \subset \mathbb{R}^n, \\ u = g, & \text{sur} \quad \partial\Omega. \end{cases} \tag{8.1}$$

8.2 RÉSOLUTION PAR DIFFÉRENCES FINIES EN 1D

En 1D le problème ci-dessus devient

$$\begin{cases} -u''(x) + c(x)u(x) = f(x), & \text{pour} \quad x \in]a, b[, \\ u(a) = u_a, \\ u(b) = u_b. \end{cases} \tag{8.2}$$

Ce problème modélise par exemple le fléchissement d'une poutre de longueur $b - a$, de section constante et de module de raideur $c(x)$, étirée suivant son axe et soumise à une charge transversale $f(x)dx$ par unité de longueur dx.

On discrétise l'intervalle $[a, b]$ avec un pas uniforme $h = \dfrac{b-a}{n+1}$ sur $n + 2$ points $x_i = a + ih$ pour $i = 0, ..., n + 1$. On note U le vecteur formé par l'approximation de la solution $u(x)$ aux points x_i. On pose $U_0 = u_a$ et $U_{n+1} = u_b$. La discrétisation par différences finies de ce problème (cf. 1) conduit au système linéaire

$$(S) \qquad A_h U = B_h,$$

où A_h est une matrice $n \times n$ tridiagonale avec

$$A_h^0 = \frac{1}{h^2} \begin{pmatrix} 2 + h^2 c_1 & -1 & 0 & \dots & \dots & 0 \\ -1 & 2 + h^2 c_2 & \ddots & \ddots & \ddots & \vdots \\ 0 & \ddots & 2 + h^2 c_3 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & 2 + h^2 c_{n-1} & -1 \\ 0 & \dots & \dots & 0 & -1 & 2 + h^2 c_n \end{pmatrix},$$

où $c_i = c(x_i)$ et B_h un vecteur de $\mathbb{R}^n$

$$B_h = \begin{pmatrix} f(a + h) + \dfrac{u_a}{h^2} \\ f(a + 2h) \\ . \\ . \\ f(b - h) + \dfrac{u_b}{h^2} \end{pmatrix}.$$

8.3 MÉTHODE DE SCHWARZ EN 1D

On découpe le domaine de calcul $[a, b]$ en deux sous-domaines avec recouvrement : on choisit n impair et deux entiers symétriques par rapport à $\frac{n+1}{2}$ tels que $i_g < \frac{n+1}{2} < i_d$. On pose $x_g = i_g h$ et $x_d = i_d h$, les deux sous-intervalles $]a, x_d[$ et $]x_g, b[$ ont bien un recouvrement non vide ($]a, x_d[\cap]x_g, b[\neq \emptyset$). On envisage maintenant de calculer la solution u du problème (8.2) en résolvant deux problèmes posés sur les deux sous intervalles $]a, x_d[$ et $]x_g, b[$.

$$(P_1) \quad \begin{cases} -u_1''(x) + c(x)u_1(x) = f(x), & \text{pour} \quad x \in]a, x_d[\\ u_1(a) = u_a, \\ u_1(x_d) = \alpha. \end{cases}$$

et

$$(P_2) \quad \begin{cases} -u_2''(x) + c(x)u_2(x) = f(x), & \text{pour} \quad x \in]x_g, b[\\ u_2(x_g) = \beta, \\ u_2(b) = u_b. \end{cases}$$

La solution u_1 (respectivement u_2) doit être la restriction sur l'intervalle $]a, x_d[$ (resp. $]x_g, b[$) de la solution u du problème posé sur l'intervalle tout entier. Les deux solutions u_1 et u_2 doivent donc être égales sur l'intervalle de recouvrement $]x_g, x_d[$ ce qui permet de déterminer les conditions aux limites en x_g et x_d :

$$u_1(x_d) = \alpha = u_2(x_d) \quad \text{et} \quad u_2(x_g) = \beta = u_1(x_g).$$

Comme on ne connaît pas a priori les valeurs de α et β, on résout ces problèmes de manière itérative : on choisit α de manière arbitraire, par exemple par interpolation linéaire des conditions aux limites

$$\alpha = \frac{1}{b-a}\left[u_a(b - x_d) + u_b(x_d - a)\right],$$

puis on pose $u_2^0(x_d) = \alpha$ et on calcule pour $k = 1, 2, \ldots$ les solutions u_1^k et u_2^k des problèmes

$$(P_1) \quad \begin{cases} -u_1''(x) + c(x)u_1(x) = f(x), & \text{pour} \quad x \in]a, x_d[\\ u_1(a) = u_a, \\ u_1(x_d) = u_2^{k-1}(x_d). \end{cases}$$

puis

$$(P_2) \quad \begin{cases} -u_2''(x) + c(x)u_2(x) = f(x), & \text{pour} \quad x \in]x_g, b[\\ u_2(x_g) = u_1^k(x_g), \\ u_2(b) = u_b. \end{cases}$$

On admet que cet algorithme converge vers la solution u du problème initial (8.2).

Discrétisation

On résout les problèmes P_1 et P_2 par différences finies, comme le problème global (8.2) du paragraphe précédent. On a choisi les bornes x_g et x_d telles que les deux sous-domaines soient de même taille. En notant V^k (respectivement W^k) le vecteur de la

solution discrète approchée sur le sous-domaine $]a, x_d[$ (resp. $]x_g, b[$) à l'itération k, l'algorithme s'écrit

$$\left\|\begin{array}{l} \text{initialisation :} \quad W_{i_d}^0 = \alpha \\ \text{pour} \quad k = 1, 2, \ldots, \quad \text{faire} \\ \quad A_{h,g} V^k = B_{h,g} + \frac{1}{h^2}[u_a, 0, ..., 0, W_{i_d}^{k-1}]^T, \\ \quad A_{h,d} W^k = B_{h,d} + \frac{1}{h^2}[V_{i_g}^k, 0, ..., 0, u_b]^T. \\ \text{fin de} \quad k \end{array}\right. \tag{8.3}$$

où $A_{h,g}$ (resp. $A_{h,d}$) est la matrice de discrétisation de l'opérateur $-\Delta + cId$ sur $]a, x_d[$ (resp. $]x_g, b[$) dans $\mathbb{R}^{i_d-1} \times \mathbb{R}^{i_d-1}$

$$A_{h,g} = \frac{1}{h^2}\begin{pmatrix} 2+h^2c_1 & -1 & 0 & \ldots & \ldots & 0 \\ -1 & 2+h^2c_2 & -1 & 0 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & 0 & -1 & 2+h^2c_{i_d-2} & -1 \\ 0 & \ldots & \ldots & 0 & -1 & 2+h^2c_{i_d-1} \end{pmatrix},$$

$$\text{et} \quad A_{h,g} = \frac{1}{h^2}\begin{pmatrix} 2+h^2c_{i_g+1} & -1 & 0 & \ldots & \ldots & 0 \\ -1 & 2+h^2c_{i_g+2} & -1 & 0 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & 0 & -1 & 2+h^2c_{n-1} & -1 \\ 0 & \ldots & \ldots & 0 & -1 & 2+h^2c_n \end{pmatrix}.$$

Les vecteurs B_g et B_d contiennent les valeurs du second membre f aux points de discrétisation

$$\left(B_{h,g}\right)_i = f(x_i) \quad \text{pour} \quad i = 1, ..., i_d - 1 \quad \text{et} \quad \left(B_{h,d}\right)_i = f(x_{i+i_g}) \quad \text{pour} \quad i = 1, ..., n - i_g.$$

Le test d'arrêt des itérations porte sur l'écart entre les valeurs calculées par les deux sous problèmes dans la zone de recouvrement $]x_g, x_d[$:

$$\|e^k\| \leqslant \varepsilon, \quad \text{avec} \quad e_{i-i_g}^k = V_i^k - W_{i-i_g}^k, \quad \text{pour} \quad i = i_g + 1, ...i_d - 1$$

Pour tester les performances de la méthode on peut aussi comparer les résultats avec la solution U calculée de la manière classique sur tout le domaine au paragraphe 8.2.

On calcule pour cela deux vecteurs e_g^k et e_d^k de composantes

$$\begin{aligned} (e_g^k)_i &= U_i - V_i^k, \quad \text{pour} \quad i = 1, ..., i_d - 1, \\ \text{et} \quad (e_d^k)_{i-i_g} &= U_i - W_{i-i_g}^k, \quad \text{pour} \quad i = i_g + 1, ..., n, \end{aligned}$$

et on observe la diminution de leur norme au cours des itérations, en même temps que celle de e_k.

Exercice 8.1

Écrire un programme mettant en œuvre l'algorithme 8.3. Afficher dans la même fenêtre graphique avec des couleurs différentes les solutions V^k, W^k et U en rafraîchissant la figure à chaque itération k. On obtient une succession de graphes comme sur la figure 8.2. Représenter l'évolution des trois erreurs $\|e^k\|$, $\|e_g^k\|$ et $\|e_d^k\|$ en fonction de k dans une autre fenêtre graphique comme sur la figure 8.1.

La solution de cet exercice se trouve en page 182

Exercice 8.2

Modifier le programme de l'exercice 8.1 pour en faire une fonction recevant en argument le nombre de points $r = i_d - i_g - 1$ dans le recouvrement et renvoyant le nombre d'itérations nécessaire pour atteindre l'erreur tolérée et le temps de calcul. Appeler la fonction pour plusieurs valeurs de r et analyser l'influence de la taille du recouvrement sur la convergence de l'algorithme.

Les figures 8.1 et 8.2 illustrent les résultats pour une poutre de 1 m de long, avec une constante de raideur constante $c = 10$. L'extrémité gauche est fixée 10 cm plus haut que la droite. La poutre est soumise à son propre poids 1 N/m ainsi qu'à une surcharge de 9 N/m sur 40 cm à partir de 20 cm du bord gauche.

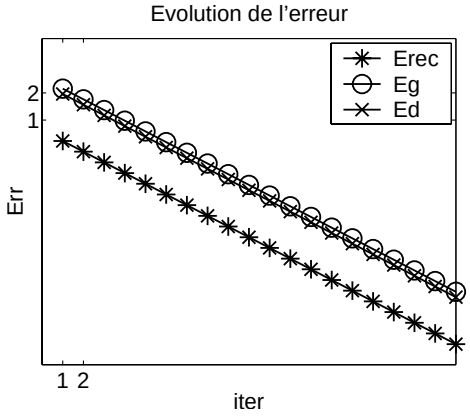


Figure 8.1 Logarithme de l'erreur en norme L^2 en fonction des itérations

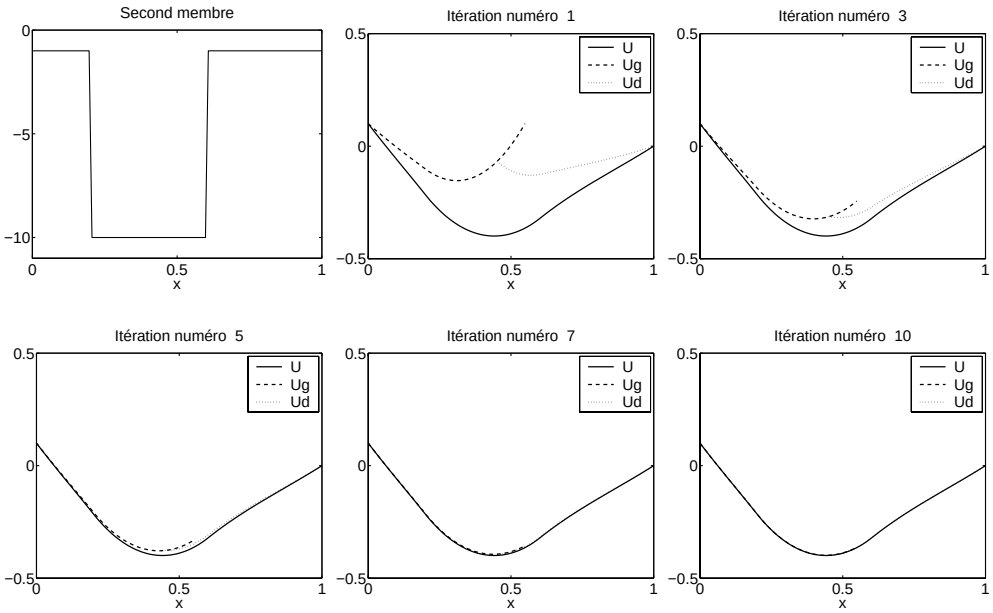


Figure 8.2 Second membre, solutions globale et sur chaque domaine pour différentes itérations

8.4 EXTENSION AU CAS 2D

On s'intéresse maintenant au problème 2D (8.1) posé sur un rectangle. On se limite au cas où $c = 0$, modélisant ainsi un problème de conduction de la chaleur dans un objet dont une des dimensions est beaucoup plus grande que les deux autres, et dans la direction de laquelle on négligera les variations. On traite en premier le cas de conditions aux limites de Dirichlet non-homogènes.

$$\begin{cases} -\Delta u(x_1, x_2) = F(x_1, x_2), \quad \text{pour } (x_1, x_2) \in]a_1, b_1[\times]a_2, b_2[, \\ u(a_1, x_2) = f_2(x_2), \quad \text{pour } x_2 \in]a_2, b_2[, \\ u(b_1, x_2) = g_2(x_2), \quad \text{pour } x_2 \in]a_2, b_2[, \\ u(x_1, a_2) = f_1(x_1), \quad \text{pour } x_1 \in]a_1, b_1[, \\ u(x_1, b_2) = g_1(x_1), \quad \text{pour } x_1 \in]a_1, b_1[, \end{cases}$$

D'un point de vue pratique, ce calcul modélise par exemple un test de choc thermique sur une poutre métallique (voir figure 8.3). On imposera par exemple une température nulle sur les faces $x_1 = a_1$ et $x_1 = b_1$, c'est-à-dire $f_2(x_2) = g_2(x_2) = 0$, une température de 50 °C sur la face $x_2 = a_2$, c'est-à-dire $f_1(x_1) = 50$ et enfin une température de 100 °C sur la face $x_2 = b_2$, soit $g_1(x_1) = 100$. Enfin il n'y a pas de source de chaleur interne, donc le second membre $F(x_1, x_2) = 0$.

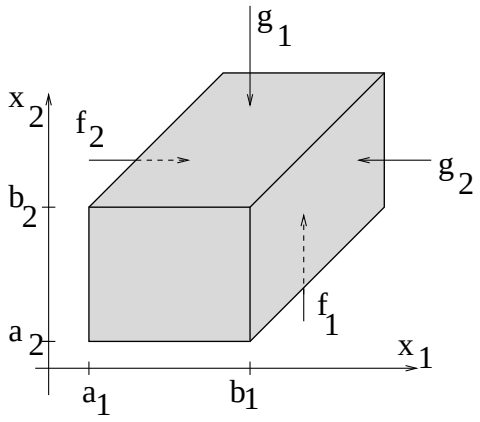


Figure 8.3 Schéma de la pièce soumise à un choc thermique

8.4.1 Résolution par différences finies en 2D

On se placera ici dans le cas où les dimensions du domaine permettent d'avoir le même pas de discrétisation dans les deux directions x_1 et x_2. C'est-à-dire :

$$h = \frac{b_1 - a_1}{n_1} = \frac{b_2 - a_2}{n_2}.$$

Sur cette grille régulière (voir figure 8.4), on note $u_{i,j} = u(a_1 + ih, a_2 + jh)$ (respectivement $f_{i,j}$) la discrétisation de la fonction $u(x_1, x_2)$ (resp. $F(x_1, x_2)$) aux points de la grille.

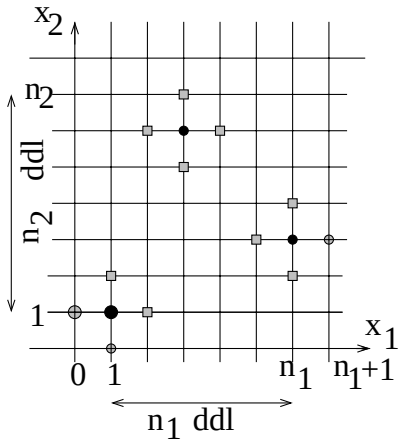


Figure 8.4 Discrétisation de l'intérieur du domaine Ω sur $n_1 \times n_2$ points

Dans ce cas, en utilisant des développements de Taylor en x_1 et en x_2 pour approcher les dérivées partielles dans chaque direction, on peut montrer que la discrétisation du Laplacien par le schéma à 5 points

$$\Delta u_{i,j} \approx \frac{4u_{i,j} - u_{i-1,j} - u_{i+1,j} - u_{i,j-1} - u_{i,j+1}}{h^2}.$$

est un $O(h^2)$ si u est de classe C^4. La solution approchée aux différences finies W est donc solution du système linéaire

$$\frac{4u_{i,j} - u_{i-1,j} - u_{i+1,j} - u_{i,j-1} - u_{i,j+1}}{h^2} + c_{i,j}u_{i,j} = f_{i,j}, \tag{8.4}$$

où les inconnues sont les $u_{i,j}$ pour $i = 1, ..., n_1$ et $j = 1, ..., n_2$. En effet les valeurs de la fonction sur les bords, correspondant aux indices $i = 0$ ou $i = n_1 + 1$ et $j = 0$ ou $j = n_2 + 1$ sont fixées par les conditions aux limites

$$u_{0,j} = f_2(a_2 + jh), \quad u_{n+1,j} = g_2(a_2 + jh), \quad u_{i,0} = f_1(a_1 + ih), \quad u_{i,n+1} = g_1(a_1 + ih).$$

Chaque ligne du système linéaire (8.4) compte au plus 5 termes non nuls : le terme diagonal a le coefficient $\frac{4}{h^2}$ et les termes extra diagonaux correspondants aux voisins

$$(i + 1, j), (i - 1, j), (i, j - 1), (i, j + 1),$$

qui ne font pas partie du bord ont le coefficient $\frac{-1}{h^2}$. Ces nœuds sont représentés par des carrés sur la figure 8.4). On peut en fait construire la matrice par blocs : les degrés de liberté (i, j), $(i + 1, j)$, et $(i - 1, j)$ sont à la fois voisins et consécutifs dans la numérotation globale des degrés de liberté. Les coefficients du système linéaire reliant les nœuds d'une même rangée j, pour $j = 1, ..., n_2$ sont donc ceux d'une matrice tridiagonale $T_j = T + D_j$, avec

$$T = \frac{1}{h^2}\begin{pmatrix} 4 & -1 & 0 & \dots & \dots & 0 \\ -1 & 4 & -1 & \ddots & \ddots & \vdots \\ 0 & \ddots & 4 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & 4 & -1 \\ 0 & \dots & \dots & 0 & -1 & 4 \end{pmatrix}, \quad D_j = \begin{pmatrix} c_{1,j} & 0 & \dots & 0 \\ 0 & c_{2,j} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & c_{n_1,j} \end{pmatrix}. \tag{8.5}$$

Les deux autres voisins du nœud (i, j) sont les nœuds $(i, j - 1), (i, j + 1)$ qui sont distants de n_1 de part et d'autre du nœud central dans la numérotation globale et leur connection est assurée par une matrice diagonale $D = \frac{-1}{h^2} I_{n_1 \times n_1}$ de part et d'autre de la matrice T_j. La matrice du système linéaire global $A\,U = B$, est donc une matrice

tridiagonale par blocs de taille $n_2 \times n_2$, chaque bloc étant de taille $n_1 \times n_1$,

$$A = \begin{pmatrix} T_1 & D & 0 & \dots & 0 \\ D & T_2 & D & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & D & T_{n_2-1} & D \\ 0 & \dots & 0 & D & T_{n_2} \end{pmatrix} \tag{8.6}$$

Le second membre B du système linéaire est un vecteur de taille $n_1\, n_2$ qu'on construit sous la forme d'une matrice $n_1 \times n_2$ pour profiter de la numérotation des degrés de liberté associée à la représentation de la grille. On commence par initialiser le vecteur second membre avec la valeur de la fonction second membre $F(x_1, x_2)$ aux nœuds de la grille.

$$B_{i,j} = f_{i,j}, \quad \text{pour} \quad 1 \leqslant i \leqslant n_1, \quad 1 \leqslant j \leqslant n_2.$$

Si le nœud (i, j) a un voisin sur le bord du maillage (représenté par un cercle grisé sur la figure 8.4), la contribution $\dfrac{u_{i',j'}}{h^2}$ de ce voisin (i', j') dans la discrétisation du Laplacien au point (i, j) est reportée au second membre, la valeur de $u_{i',j'}$ étant donnée par les conditions aux limites. Donc on rajoute la contribution des conditions aux limites sur tous les termes $B_{1,i}$, $B_{n_1,i}$ pour $i = 1, \dots, n_2$ et $B_{i,1}$, B_{i,n_2} pour $i = 1, \dots, n_1$. Attention aux quatre coins !...

$$B_{1,i} = B_{1,i} + \frac{f_2(a_2 + ih)}{h^2}, \quad B_{n_1,i} = B_{n_1,i} + \frac{g_2(a_2 + ih)}{h^2}, \quad \text{pour} \quad 1 \leqslant i \leqslant n_1,$$

$$B_{i,1} = B_{i,1} + \frac{f_1(a_1 + ih)}{h^2}, \quad B_{i,n_2} = B_{i,n_2} + \frac{g_1(a_1 + ih)}{h^2}, \quad \text{pour} \quad 1 \leqslant j \leqslant n_2..$$

Pour la mise en œuvre numérique on propose de prendre $c(x) = 0$ dans un premier temps.

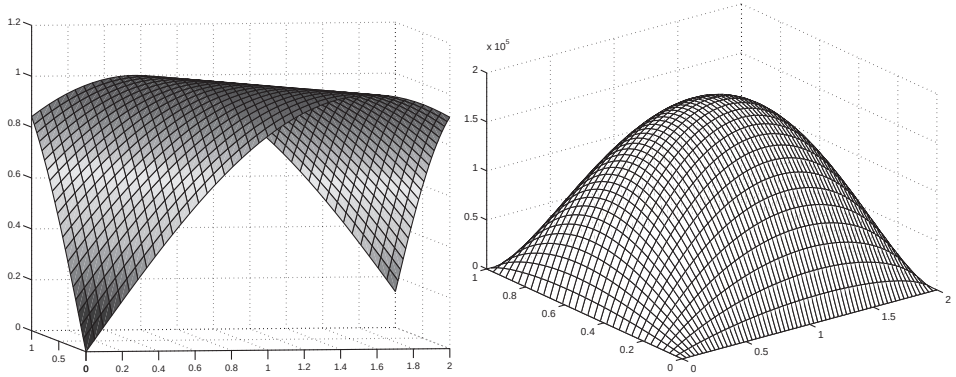


Figure 8.5 Solution globale (à gauche) et erreur entre la solution exacte et l'approximation par différences finies (à droite).

Exercice 8.3

1. Écrire une fonction `LaplaceDirichlet` pour calculer la matrice A de taille $n_1 n_2 \times n_1 n_2$ du système linéaire global. La fonction renvoie la matrice A. Elle reçoit en argument les bornes inférieures du domaine, a_1, a_2, le nombre de points dans chaque direction n_1 et n_2, et le pas h. Construire la matrice A par blocs en utilisant une sous matrice triangulaire T et la matrice identité de taille $n_1 \times n_1$.
2. Écrire une fonction `SecondMembre2d` pour calculer le second membre du système linéaire. La fonction renvoie le vecteur B. Elle reçoit en argument les bornes inférieures du domaine, a_1, a_2, le nombre de points dans chaque direction n_1 et n_2, et le pas h. Elle fait appel à la fonction `sm2d(x1,x2)` qui calcule les valeurs de la fonction second membre $F(x_1, x_2)$.
3. Écrire une fonction `CondLim2d` pour prendre en compte dans le second membre les conditions de Dirichlet non homogènes.
4. Écrire une fonction `DifFin2d` pour calculer la solution par différences finies. La séquence d'appel sera

```
function [n2,b2,Solm]=DifFin2d(n1,a1,a2,b1,b2, sm2d,f1,g1,
                          f2,g2,...SecondMembre2d,Laplace)
```

5. On choisit une fonction $u(x_1, x_2) = \sin(x_1 + x_2)$ qui est la solution exacte du problème $-\Delta u = f$ avec la fonction second membre est donc :

$$F(x_1, x_2) = 2\sin(x_1 + x_2).$$

   Les conditions aux limites sont les restrictions de la solution exacte sur les bords :

$$f_1(x_1) = \sin(x_1 + a_2), \qquad g_1(x_1) = \sin(x_1 + b_2),$$
$$f_2(x_2) = \sin(a_1 + x_2), \qquad g_2(x_2) = \sin(b_2 + x_2)$$

   Programmer les fonctions `sm2dExact(x1,x2)`, `f1Exact(x1)`, `g1Exact(x1)`, `f2Exact(x2)`, `g2Exact(x2)` correspondant à ce cas test, ainsi que la fonction `u2dExact(x1,x2)` pour calculer la solution exacte aux nœuds la grille.
6. Écrire un programme `TestDifFin2d` pour tester les deux fonctions précédentes : pour cela on prendra soin de bien avoir le même pas de discrétisation dans les deux directions, en ajustant par exemple la dimension du domaine dans la direction x_2. Pour les paramètres $a_1 = a_2 = 0$, $b_1 = 1$, $b_2 = 2$, $n_1 = 20$, on obtient la solution représentée à gauche sur la figure 8.5. On vérifiera les calculs en représentant graphiquement l'erreur, c'est-à-dire la différence entre la solution exacte $u(x_1, x_2) = \sin(x_1 + x_2)$, et la solution différences finies, comme sur le graphe de droite de la figure 8.5.

7. Adapter le programme précédent au cas du choc thermique, pour obtenir la solution représentée sur la figure 8.6, d'abord avec une section carrée $b_1 - a_1 = b_2 - a_2 = 6$, puis rectangulaire $b_1 - a_1 = 6$ et $b_2 - a_2 = 20$.

La solution de cet exercice se trouve en page 183

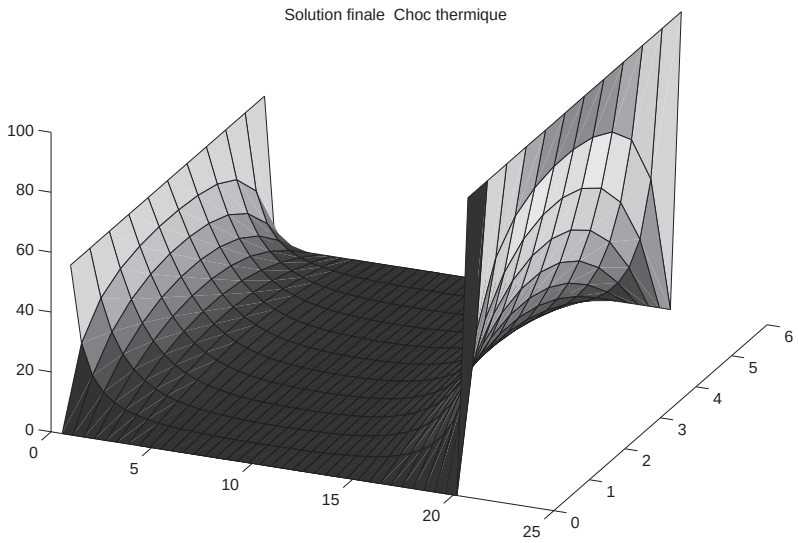


Figure 8.6 Solution du problème de choc thermique.

8.4.2 Décomposition de domaine dans le cas 2D

On va maintenant appliquer la technique du paragraphe 8.3 dans le cas 2D. Le domaine va être décomposé en s sous-domaines avec recouvrement dans la direction x_2. Là encore pour simplifier la mise en œuvre informatique on va supposer qu'on peut discrétiser le domaine avec le même pas h dans les deux directions. De plus on va aussi imposer que les sous-domaines ont tous la même taille $(n_2 + 1)h$ et que les recouvrements ont tous le même nombre de mailles r. La figure 8.7 montre un exemple de décomposition vérifiant ces contraintes. Noter que l'on ne peut pas forcément les respecter pour un domaine $[a_1, b_1] \times [a_2, b_2]$ quelconque, et qu'il sera éventuellement nécessaire d'ajuster la longueur totale $b_2 - a_2$.

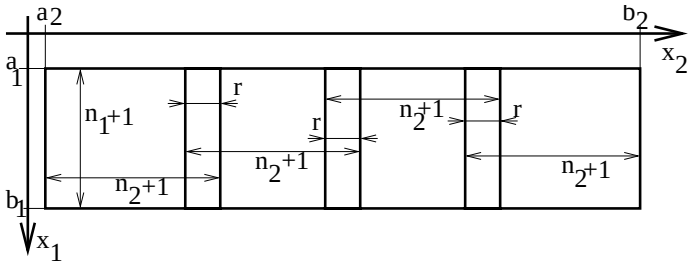


Figure 8.7 Décomposition avec recouvrement constant, en 4 sous-domaines identiques

On note $u^{k,p}$ la solution sur le sous-domaine k $[a_1, b_1] \times [a_2^k, b_2^k]$, à l'itération p, pour $k = 1, ..., s$ et $p = 0, 1, ...$ Les limites a_2^k et b_2^k sont égales à $a_2^1 = a_2$ et $a_2^k = a_2^{k-1} + (n_2 + 1 - r)h$ pour $k > 1$ et $b_2^k = a_2^k + (n_2 + 1)h$.

Avec ces notations, $u^{k,p}$ est solution du problème

$$
\left\{
\begin{array}{l}
-\Delta u^{k,p}(x_1, x_2) = F(x_1, x_2), \quad \text{pour } (x_1, x_2) \in]a_1, b_1[\times]a_2^k, b_2^k[, \\
u^{k,p}(a_1, x_2) = f_2(x_2), \quad \text{pour } x_2 \in]a_2^k, b_2^k[, \\
u^{k,p}(b_1, x_2) = g_2(x_2), \quad \text{pour } x_2 \in]a_2^k, b_2^k[, \\
u^{k,p}(x_1, a_2^k) = \left\{ \begin{array}{ll} f_1(x_1), & \text{pour } \quad k = 1 \\ u^{k-1,p}(x_1, a_2^k), & \text{pour } \quad k = 2, ..., s \end{array} \right. \quad \text{pour } \quad x_1 \in]a_1, b_1[, \\
u^{k,p}(x_1, b_2^k) = \left\{ \begin{array}{ll} g_1(x_1), & \text{pour } \quad k = s \\ u^{k+1,p-1}(x_1, b_2^k), & \text{pour } \quad k = 1, ..., s-1 \end{array} \right. \quad \text{pour } x_1 \in]a_1, b_1[.
\end{array}
\right.
$$

À la première itération la condition aux limites sur le bord de droite d'un sous-domaine autre que le dernier n'est pas définie, on la fixe arbitrairement, par exemple en interpolant linéairement les deux conditions f_1 et g_1 :

$$u^{k+1,0}(x_1, b_2^k) = ((b_2 - b_2^k) f_1(x_1) + (b_2^k - a_2) g_1(x_1))/(b_2 - a_2).$$

On arrête les itérations quand la somme (ou le maximum) des normes des erreurs sur chaque recouvrement est inférieure à un seuil toléré.

On note X_1 le vecteur des abscisses $a_1 + jh$, $j = 1, \ldots, n_1$ et X_2^i le vecteur des ordonnées dans le i^{eme} sous-domaine $a_2 + jh + (i-1)(n2 + 1 - r)$, $j = 1, \ldots, n_2$. L'algorithme de Schwarz en 2D s'écrit

$$
\begin{array}{l}
\text{initialisation :} \\
\quad U_{a_2 g} = f_1(X_1), \quad U_{b_2 d} = g_1(X_1) \\
\quad U^{i,0}_{.,r} = ((b_2 - b_2^i) U_{b_2 d} + (b_2^i - a_2) U_{a_2 g})/(b_2 - a_2)), \quad \text{pour} \quad i = 2, \ldots, s \\
\quad U^i_{a_1} = f_1(X_2^i), \quad U^i_{b_1} = g_1(X_2^i), \quad B^i_{.,.} = F(X_1, X_2^i), \\
\quad B^i_{1,.} = B^i_{1,.} + U^i_{a_1}/h^2, \quad B^i_{n_1,.} = B^i_{n_1,.} + U^i_{b_1}/h^2 \\
\text{pour} \quad p = 1, 2, \ldots \quad \text{faire} \\
\quad \text{pour} \quad i = 1, \ldots, s \quad \text{faire} \\
\qquad \text{si } i = 1, \quad U^i_{a_2} = U_{a_2 g} \quad \text{sinon} \quad U^i_{a_2} = U^{i-1,p}_{.,n_2+1-r} \\
\qquad \text{si } i = s, \quad U^i_{b_2} = U_{b_2 d} \quad \text{sinon} \quad U^i_{b_2} = U^{i+1,p-1}_{.,r} \\
\qquad B^{i,p} = B^i, \quad B^{i,p}_{.,1} = B^{i,p}_{.,1} + U^i_{a_2}/h^2 \quad B^{i,p}_{.,n_2} = B^{i,p}_{.,n_2} + U^i_{b_2}/h^2 \\
\qquad \text{résoudre } A U^{i,p} = B^{i,p} \\
\qquad \text{si } i > 1, \quad R^i_{.,j} = U^{i,p}_{.,j} - U^{i-1,p}_{.,n2-r+1+j}, \quad \text{pour } j = 1, \ldots, r-1 \\
\quad \text{fin de} \quad i \\
\quad E^p = \sup_{i=2,\ldots,s} \|R^i\| \\
\text{si } E^p < \varepsilon \quad \text{fin de} \quad p
\end{array}
\tag{8.7}
$$

où A est la matrice de discrétisation de l'opérateur $-\Delta$, définie par (8.6). Elle est ici identique pour tous les sous-domaines.

Exercice 8.4

1. Écrire une fonction `Schwarz2d` mettant en œuvre cet algorithme. La syntaxe d'appel sera

```
function [conviter,cpu,mem,n2,b2]=Schwarz2d(n1,s,r,a1,
                  a2,b1,b2,... sm2d,f1,g1,f2,g2,
                  SecondMembre2d,Laplace,n11)
```

   dont les paramètres sont
   – `n1`, le nombre de mailles dans la direction x_1,
   – `n11`, le nombre de degrés de liberté dans la direction x_1,
   – `s`, le nombre de sous-domaines,
   – `r`, le nombre de mailles dans les recouvrements dans la direction x_2, – `Laplace`, le nom de la fonction pour calculer la matrice du Laplacien,
   – `SecondMembre2d`, le nom de la fonction pour calculer le second membre,
   – `sm2d`, le nom de la fonction $F(x_1, x_2)$,
   – `f1`, le nom de la fonction $f1(x_1)$ condition limite sur le bord $x_2 = a_2$,
   – `g1`, le nom de la fonction $g1(x_1)$ condition limite sur le bord $x_2 = b_2$,
   – `f2`, le nom de la fonction $f2(x_2)$ condition limite sur le bord $x_1 = a_1$,
   – `g2`, le nom de la fonction $g2(x_2)$ condition limite sur le bord $x_1 = b_1$,

   et renvoyant comme argument de sortie
   – `conviter`, le nombre d'itérations nécessaires pour que l'erreur maximale dans les recouvrements soit inférieure à la tolérance spécifiée `tol`,
   – `cpu`, le temps de calcul,
   – `mem`, la mémoire nécessaire.
2. Écrire un programme `TestSchwarz2d` pour tester l'algorithme avec la même fonction f que dans le cas global pour les valeurs suivantes des paramètres $a_1 = a_2 = 0$, $b_1 = 1$, $n_1 = 9$, $b_2 = 30$, $r = 10$, $s = 20$.

La solution de cet exercice se trouve en page 184.

Étude des performances de la méthode : on voudrait maintenant étudier l'influence de la taille des sous-domaines sur la vitesse de convergence. Pour cela on va évaluer pour une décomposition en sous-domaines donnée, le temps de calcul pour atteindre la précision requise à l'aide des commandes `tic` et `toc`. Pour comparer des choses comparables, il faut garder le maximum de paramètres constants, en particulier comparer des performances pour des décompositions du même domaine global ce qui impose des contraintes sur le nombre de sous-domaines, leur taille, et la taille du recouvrement.

Exercice 8.5

Fixer $b_1 = 1$, $b_2 = 50$, $n_1 = 9$, et la taille du recouvrement $r = 4$. Balayer la plage des valeurs réalistes pour le nombre s de sous-domaines - par exemple de 5 à 60 - et pour chaque valeur de s pour laquelle la décomposition est possible, calculer la solution en appelant la fonction `Schwarz2`. Représenter les performances en temps de calcul, en mémoire, et en nombre d'itérations, en fonction des paramètres `r` et `s`. Analyser l'influence de la taille du recouvrement sur la convergence de l'algorithme.

La solution de cet exercice se trouve en page 188.

8.4.3 Mise en œuvre de conditions limites réalistes

Des cas tests plus réalistes de calculs de conduction de la chaleur impliquent la mise en œuvre de conditions limites autres que Dirichlet. On peut par exemple calculer la répartition de température dans une barre bus, comme celle schématisée sur la figure 8.8 dans laquelle le courant électrique génère de la chaleur de manière uniforme à raison de $F(x_1, x_2) = q = 10^6$ W/m^3.

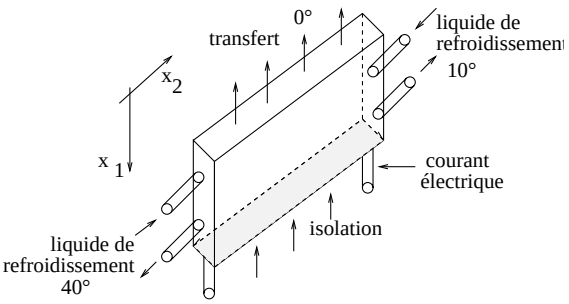


Figure 8.8 Schéma d'une barre bus

La température est imposée sur les faces des électrodes - 40 °C à gauche et 10 °C à droite, grâce à la circulation d'un liquide de refroidissement. Les deux faces latérales de la barre ainsi que la face inférieure sont isolées, c'est-à-dire qu'on y impose une condition limite de type Neumann. Sur la face supérieure en revanche, on impose une condition limite de type Fourier pour modéliser le refroidissement par convection naturelle, avec un coefficient de transfert thermique égal à $c_{th} = 75$ W/m^2 et une température extérieure qu'on prendra égale à 0 °C. Le coefficient de diffusivité thermique de l'alliage est égal à $k = 20$ W/m K. Le problème (8.1) devient dans ce cas particulier

$$\begin{cases} -k\Delta u(x_1, x_2) = q, \quad \text{pour} \quad x \in \Omega, \\ u = 40, \quad \text{sur} \quad x_2 = a_2 \\ u = 10, \quad \text{sur} \quad x_2 = b_2 \\ \dfrac{\partial u}{\partial n} = 0, \quad \text{sur} \quad x_1 = b_1 \\ \dfrac{\partial u}{\partial n} + c_{th}(u - u_{ext}) = 0, \quad \text{sur} \quad x_1 = a_1. \end{cases} \tag{8.8}$$

Ici ∂n désigne la dérivée par rapport à la direction normale à la surface. Pour discrétiser ces conditions aux limites de type Neumann et Fourier, on réintroduit les degrés de liberté correspondants aux nœuds sur les bords $x_1 = a_1$ et $x_1 = b_1$. On aura donc maintenant $N_1 + 2$ nœuds dans cette direction. On écrit d'une part pour la condition de Neumann

$$T_{N_1+1,j} = T_{N_1+2,j},$$

ce qui permet d'éliminer la référence au nœud extérieur $T_{N_1+2,j}$ dans la discrétisation du Laplacien aux points d'indices $(N_1 + 1, j)$, ce qui donne finalement

$$\frac{3u_{N_1+1,j} - u_{N_1,j} - u_{N_1+1,j-1} - u_{N_1+1,j+1}}{h^2} = f_{N_1+1,j}.$$

D'autre part la discrétisation de la condition de Fourier donne

$$u_{0,j} - u_{-1,j} + hc_{th}u_{0,j} = 0,$$

ce qui permet d'éliminer la référence aux points $-1, j$ dans la discrétisation du Laplacien aux points d'indices $(0, j)$, ce qui donne finalement

$$\frac{3u_{0,j} + hc_{th}u_{0,j} - u_{1,j} - u_{0,j-1} - u_{0,j+1}}{h^2} = f_{0,j}.$$

Exercice 8.6

1. En s'inspirant de l'algorithme 8.7, écrire l'algorithme de Schwarz avec les conditions limites de Fourier et de Neumann.
2. Écrire une fonction `LaplaceFourier` pour construire la matrice tridiagonale par blocs du système linéaire.
3. Écrire les fonctions `SecondMembre2dFourier`, `f1`, `g1`, `f2`, `g2`, et `sm2d` pour le problème de la barre bus.
4. Modifier la fonction `DifFin2d` et le programme `TestDifFin2d` pour pouvoir traiter ce problème avec l'algorithme différences finies global.
5. Modifier la fonction `Schwarz2d` et le programme `TestSchwarz2d` pour pouvoir traiter ce problème avec l'algorithme de Schwarz.

La solution de cet exercice se trouve en page 184 pour la solution globale et en page 189 pour la solution par décomposition de domaines.

On obtient avec l'algorithme de résolution global la solution représentée sur la figure 8.9. On voit sur le bord en $x1 = 0$ l'influence de la condition de Fourier qui fait décroître la solution vers la température extérieure.

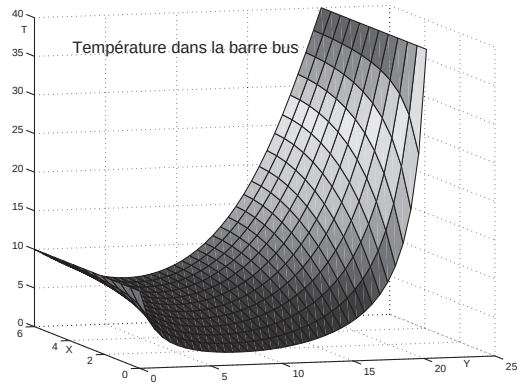


Figure 8.9 Température dans la barre bus

8.4.4 Extensions possibles

La première extension du point de vue décomposition de domaines est bien entendu d'adapter l'implémentation informatique à une décomposition en sous-domaines de tailles différentes. La gestion des degrés de liberté est alors plus compliquée et il faut calculer une matrice du système linéaire pour chaque sous-domaine. Faut-il les stocker en mémoire ou bien les recalculer à chaque itération ? Quelle est l'influence de la stratégie choisie sur le temps de calcul ?

On peut aussi donner une autre envergure au projet en s'attaquant à la mise en œuvre informatique d'une décomposition dans les deux directions, ce qui permet d'appliquer la méthode à un problème posé sur un domaine de forme complexe. Le stockage de la solution et la correspondance des degrés de liberté d'un même point dans les différents sous-domaines le contenant nécessitent une implémentation très rigoureuse.

8.5 SOLUTIONS ET PROGRAMMES

Solution de l'exercice 8.1

Le script `Schwarz1d.m` met en œuvre la méthode de Schwarz dans le cas de deux sous-domaines de même longueur. Cette contrainte simplifie l'implémentation car la matrice du système linéaire sera la même dans les deux sous-domaines. Dans le cas où le coefficient $c(x)$ est constant elle est identique dans les deux sous-domaines. Pour l'imposer, il faut être minutieux dans la traduction des indices mathématiques en MATLAB . Encore une fois, attention : les indices d'un tableau commencent forcément à 1. Une méthode consiste à fixer le nombre de pas d'espace dans le domaine total, égal à un nombre pair, donc le nombre de points, y compris les bords a et b, à

un nombre impair n_x. Le pas d'espace est noté $h = (b - a)/(n_x - 1)$. Puis on fixe le nombre – pair – de pas d'espace dans le recouvrement $2n_{rec}$. On en déduit la position du bord gauche du sous-domaine de droite :

$$x_g = 0.5 * (a + b) - n_{rec}h,$$

et la position du bord droit du sous-domaine de gauche :

$$x_d = 0.5 * (a + b) + n_{rec}h.$$

Enfin le nombre d'intervalles dans chaque sous-domaine est égal à

$$i_g = i_d = (n_x + 1)/2 + n_{rec} - 1.$$

Une fois ces paramètres définis, on peut construire la matrice $i_g - 1 \times i_g - 1$ des différences finies pour les sous-domaines. On construit aussi les deux seconds membres – sans l'influence des conditions aux limites aux points x_d et x_g qui varient à chaque itération.

Le script `Schwarz1d.m` fait appel à la fonction `sm1d.m` pour évaluer le second membre.

Solution de l'exercice 8.3

Pour l'implémentation du problème 2D, il est intéressant de conserver la double numérotation des nœuds associée au maillage cartésien 2D pour la représentation graphique de la solution, l'implémentation des conditions aux limites et le calcul du second membre. La représentation globale dans un vecteur colonne n'est indispensable qu'au moment de la résolution du système linéaire. MATLAB peut facilement convertir le tableau $n_1 \times n_2$ contenant les inconnues $u_{i,j}$ en un tableau u_k avec $k = 1, ..., n_1 \times n_2$.

```
%Si size(tab)=[n1,n2]
col=tab(:); % size(col)=[n1*n2,1] et col((j-1)*n1+i)=tab(i,j)
% inversement si  size(col)=[n1*n2,1]
tab=zeros(n1,n2);
tab(:)=col;
```

On donne d'abord des solutions de programmation MATLAB pour les fonctions : la matrice de discrétisation par différences finies du Laplacien 2D dans le cas de conditions limites de type Dirichlet est construite par la fonction `LaplaceDirichlet.m`.

Dans le cas du test proposé à la question 6, le second membre et les conditions limites correspondant à la solution exacte

$$u(x_1, x_2) = \sin(x_1 + x_2)$$

sont programmés dans les fonctions `SecondMembre2dDirichlet.m`, `sm2dExact.m`, `f1Exact.m`, `g1Exact.m`, `f2Exact.m`, `g2Exact.m`. Le calcul de la solution par différences finies, effectué dans la fonction `DifFin2dDirichlet`, reçoit en argument les fonctions `f1Exact`, `g1Exact`, `f2Exact`, `g2Exact`, et `sm2dExact` appelées localement `f1`, `g1`, `f2`, `g2`, `sm2d`, de manière à pouvoir traiter d'autres cas tests et d'autres conditions limites. Dans le cas présent des conditions de Dirichlet non homogènes sur toute la frontière, le nombre de degrés de liberté dans la direction x_1 (respectivement x_2) est égal à `n1` (resp. `n2`), le nombre de nœuds internes dans cette direction.

Dans la première partie du script d'appel `TestDifFin2d`, on traite justement le cas test de la question 6, et on compare la solution différences finies avec la solution exacte en représentant graphiquement leur différence. On calcule ensuite la solution du choc thermique 8.3, pour lequel on ne connaît pas la solution exacte, en envoyant en argument à la fonction `DifFin2dDirichlet` les fonctions `sm2dCT.m`, `f1CT.m`, `g1CT.m` et `f2CT.m` pour calculer le second membre. Enfin le dernier calcul effectué dans le script `TestDifFin2d.m` correspond au problème de la barre bus de l'exercice 8.6. Il fait appel à la fonction `DifFin2dFourier` pour lequel on programme le Laplacien dans `LaplaceFourier.m` avec la prise en compte des conditions de Neumann sur le bord $x_1 = a_1$ et de Fourier sur le bord $x_1 = b_1$, ainsi que de nouvelles fonctions pour calculer le second membre `SecondMembreFourier.m` et `sm2dBB.m` et prendre en compte les conditions limites de Dirichlet non homogènes imposées cette fois ci seulement sur les bords parallèles à x_1 à l'aide des fonctions `f1BB.m` et `g1BB.m`.

Solution de l'exercice 8.4

L'algorithme 8.7 correspondant à la méthode de Schwarz pour des conditions limites de type Dirichlet sur les quatre bords est programmé dans la fonction `Schwarz2dDirichlet` ci-dessous et testée dans le script `TestSchwarz2d` pour les deux exemples de l'exercice précédent.

Listing 8.1 *Schwarz2dDirichlet.m*

```
function [conviter,cpu,mem,n2,b2]=Schwarz2dDirichlet(n1,s,r,a1,
                    a2,b1,b2,... f,f1,g1,f2,g2,detaille)
tic    % début du compteur de temps
h=(b1-a1)/(n1+1);
n2tot=round(b2/h);          % il y a n2tot mailles au total,
n2=round((n2tot-r*(1-s))/s)-1;    % et n2+1 mailles dans chaque
                                  % sous-domaine
n2tot= s*(n2+1)+r*(1-s);
b2=a2+h*n2tot;            % au final taille rectifiée
% taille mémoire pour stocker la matrice et la solution
mem=(n1*n2)^2+s*n1*n2;
```

```
 if detaille,
  fprintf('%d sous-domaines identiques de largeur %d:\n',s,n1);  pause(1)
  fprintf('chaque sous-domaine a %d mailles\n',n2+1);
  fprintf('les recouvrements font %d mailles\n',r)
  fprintf('le total fait %f sur %d mailles\n',b2,n2tot);
  fprintf('Taille mémoire pour stocker la matrice et la solution=
    %d\n',mem);
    %
end
%
% la taille de chaque morceau sera n1 x n2
%
a11=a1-h;    % condition de Dirichlet sur le bord // à X2
%
% On prépare les conditions aux limites fixes dans des tableaux
Ua2g=feval(f1,a1+h*[1:n1])';   % condition limite sur le bord x2=a2
Ub2d=feval(g1,a1+h*[1:n1])';    % condition limite sur le bord x2=b2
% et les second membres sur chaque sous domaine auquel on ajoutera la
% contribution des bords interieurs au domaine, qui varie
% a chaque iteration
SM=zeros(n1*n2,s);
starts=0;   %indice de départ des morceaux
Smm=zeros(n1,n2);
Solm=zeros(n1,n2);
for i=1:s
  SM(:,i)=SecondMembre2dDirichlet(f,h,n1,n2,a1,a2+starts*h);
    % condition limite de Dirichlet sur le bord x1=a1
  Ua1(i,:)=feval(f2,a2+starts*h+[1:n2]*h);
    % condition limite de Dirichlet sur le bord x1=b1
  Ub1(i,:)=feval(g2,a2+starts*h+[1:n2]*h);
  Smm(1,:)=Ua1(i,:)/h^2;
  Smm(n1,:)=Ub1(i,:)/h^2;
  SM(:,i)=  SM(:,i)+Smm(:);
  Solm(:,r)=(i*Ub2d+(s-i)*Ua2g)/s;
  Solcol(:,i)=Solm(:);
  starts=starts+n2+1-r;
end
Smm=zeros(n1,n2);
Lapl=-LaplaceDirichlet(h,n1,n2);
maxiter=100; conviter=1; err=1; epsilon=0.001;
Solcol=zeros(n1*n2,s);
ERR=[];
while err>epsilon & conviter<maxiter
  if detaille,
     hold off;    pause(0.1);    fprintf('iteration numero
         %d \n',conviter);
  end
  starts=0;
  err=0;
  Ua2=Ua2g;   %bord gauche en x2 contient n1+2 lignes
```

```
  for i=1:s
    if i<s     % On récupère la condition limite sur le bord
        % droit à partir de la solution à l'itération précédente
        % sur le morceau suivant à droite
      Solmd=zeros(n1,n2);Solmd(:)=Solcol(:,i+1);
      Ub2=Solmd(:,r);
    else
      Ub2=Ub2d;   % on prend la condition limite exacte,
                  % sur le bord droit.
    end
      %
    Smm(:,1)=  Ua2/h^2;
    Smm(:,n2)= Ub2/h^2;
    Smcol=SM(:,i)+Smm(:);
    Solcol(:,i)=Lapl Smcol;
    Solm=zeros(n1,n2);Solm(:)=Solcol(:,i);
      % on récupère la condition limite à gauche pour le morceau suivant
    Ua2=Solm(:,n2-r+1);
    if i>1   % on extrait le recouvrement
      R=Solmg(:,n2-r+2:n2)-Solm(:,1:r-1);
      err=max(err,norm(R(:),inf));
    end
    Solmg=Solm;
    if detaille,
      surf(a2+h*starts+h*[1:n2],a11+h*[1:n1],Solm);
      title(strcat('itération ',int2str(conviter)))
      if i==1
        hold on
      end
    end
    starts=starts+n2+1-r;
  end
  ERR=[ERR,err];
  conviter=conviter+1;
  if detaille,
    fprintf('erreur =%f à l"itération %d\n',err,conviter)
    pause(2)
  end
end
cpu=toc;   % arrêt du compteur de temps
if detaille,   % Visualisation après convergence
  Ua2g=[feval(f1,a1); Ua2g;feval(f1,b1)];
  Ub2d=[feval(g1,a1); Ub2d;feval(g1,b1)];
  fprintf('convergence après %d itérations en %d secondes\n',
                    conviter,cpu);
  fprintf('convergence après %d itérations\n',conviter);
  figure;  hold on;
  Solmd=zeros(n1,n2);Solmd(:)=Solcol(:,1);
    % dans le cas des conditions de Dirichlet sur le bord // à x2
    % on rajoute à la solution sur chaque morceau deux lignes
    % correspondant aux conditions limites en x1=a1 et en x1=b1
```

```
  starts=0;
  Solmd= [Ua1(1,:);Solmd;Ub1(1,:)];    % dans le cas des conditions
                                       % de Dirichlet
  Solmd=[Ua2g,Solmd];   % condition limite exacte sur le bord gauche
  surf(a2+h*starts+h*[0:n2],a1+h*[0:n1+1],Solmd);
  starts   =starts+n2+1-r;
for i=2:s-1
    Solmd=zeros(n1,n2);Solmd(:)=Solcol(:,i);
    Solmd= [Ua1(i,:);Solmd;Ub1(i,:)];
    surf(a2+h*starts+h*[1:n2],a1+h*[0:n1+1],Solmd);
    starts=starts+n2+1-r;
  end
  Solmd=zeros(n1,n2);Solmd(:)=Solcol(:,s);
  Solmd= [Ua1(s,:);Solmd;Ub1(s,:)];
  Solmd=[Solmd,Ub2d];    %condition limite exacte sur le bord droit
  surf(a2+h*starts+h*[1:n2+1],a1+h*[0:n1+1],Solmd);
  starts   =starts+n2+1-r;
  title('Solution finale')
end
```

On remarque qu'on a introduit un tableau `SM` indicé par le numéro du sous-domaine pour contenir le second membre du système linéaire pour chaque sous-domaine. Ce tableau est initialisé en prenant en compte la contribution de la fonction second membre $f(x_1, x_2)$ ainsi que les conditions limites de Dirichlet sur les bords du domaine global. On utilise ensuite ce tableau au cours des itérations, dans la boucle sur les sous-domaines, pour initialiser le second membre `Smm` du système linéaire, auquel on rajoute la contribution de la condition limite de Dirichlet sur les bords intérieurs, qui dépend de la solution sur les sous-domaines voisins.
La matrice `Lapl` du système linéaire est la même pour tous les sous-domaines et ne dépend pas de la solution ; elle est donc calculée à l'extérieur de la boucle sur les itérations.

La figure 8.10 montre les solutions après 1 et 4 itérations pour le premier cas test où on connaît la solution exacte.

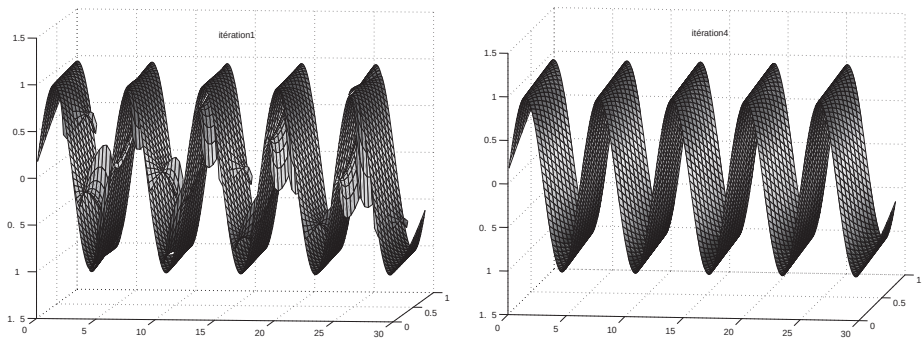


Figure 8.10 Solutions calculées sur 20 sous-domaines, après 1 et 4 itérations

Solution de l'exercice 8.5

Pour l'analyse de la convergence, on propose le script `Perf.m`, qui fait aussi appel à la fonction `Schwarz2dDirichlet`. On reprend premier le cas test, avec cette fois ci la longueur du domaine $b_2 - a_2 = 50$ alors que la largeur reste égale à 1, qu'on subdivise avec 11 mailles, soit $n_1 = 10$ degrés de liberté dans la direction x_1. On fixe la taille du recouvrement des sous-domaines égale à 10 mailles. On essaye toutes les valeurs 'raisonnables' de nombre de sous-domaines, de 5 (avec 117 ddl par sous-domaines dans la direction x_2) à 60 (avec 18 ddl par sous-domaines). Comme on impose d'avoir le même pas de discrétisation dans les deux directions x_1 et x_2, certaines configurations sont impossibles : le pas h est fixé par le nombre de points suivant x_1, $n_1 = 10$, soit $h = (b_1 - a_1)/(n_1 + 1)$. Donc le nombre de points internes dans la direction x_2 est lui aussi fixé à $n_2^{total} = (b_2 - a_2)/h$, avec la contrainte que ce nombre soit entier. Ensuite, le nombre de points par sous-domaines, n_2, en tenant compte des recouvrements, doit vérifier $s(n_2 + 1) = n_2^{total} - r(1 - s)$, et bien sûr être un nombre entier.

Pour les configurations possibles, on appelle la fonction SchwarzDirichlet, en mettant cette fois-ci le paramètre `detaille` à 0 pour ne pas avoir toutes les sorties graphiques. En revanche on récupère en paramètres de sortie le nombre d'itérations, le temps de calcul, la taille mémoire nécessaire et le nombre de points par sous domaines n_2.

La figure 8.11 montre les résultats obtenus pour deux tailles de recouvrement $r = 4$ et $r = 10$ mailles. La comparaison de l'évolution du temps de calcul avec celle du nombre d'itérations est particulièrement instructive. On s'attend en effet à ce que le nombre d'itérations augmente avec le nombre de sous-domaines, mais comme le temps de calcul nécessaire pour chaque sous-domaine diminue quand leur nombre augmente l'évolution du temps de calcul global est moins prévisible. En particulier il semble possible que le nombre optimal de sous-domaines dépende de la taille du recouvrement.

La discrétisation reste constante dans les deux directions : $h = 0,1$, de manière à pouvoir utiliser la même matrice du Laplacien dans tous les sous-domaines. Seules les décompositions conduisant à une longueur totale de 50 avec une tolérance de $0,2$ pour cent sont envisagées. A titre de comparaison, le calcul direct de la solution globale sur 9×499 degrés de liberté nécessiterait 35 secondes de temps de calcul, soit plus longtemps que la méthode par décomposition dans la pire des configurations. La mémoire nécessaire pour stocker la matrice du système global, de l'ordre de 2.10^7 est elle aussi prohibitive.

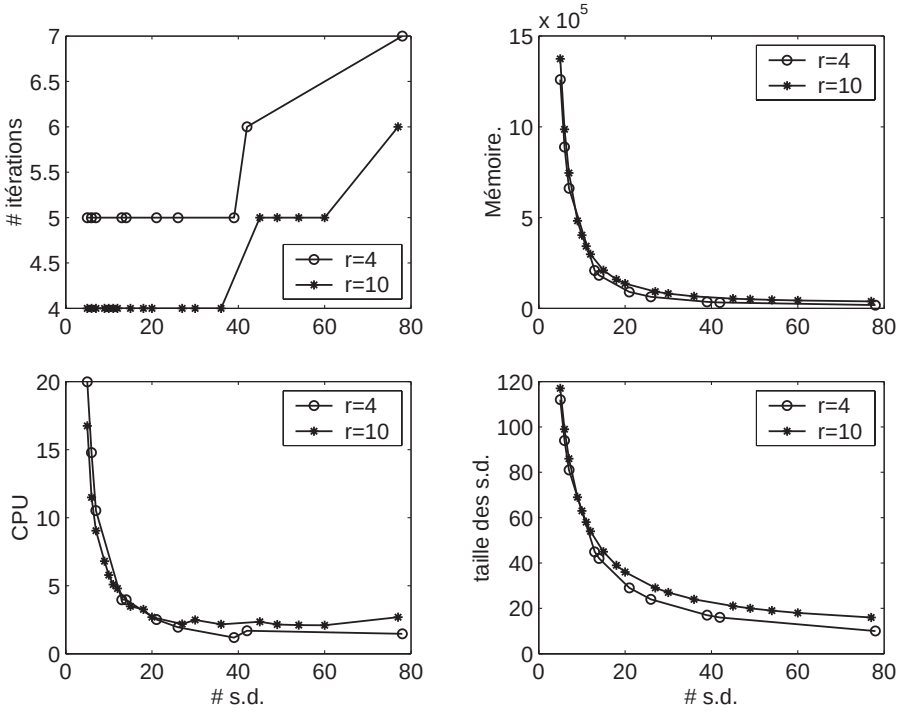


Figure 8.11 Performances de la décomposition en fonction du nombre de sous-domaines pour deux tailles de recouvrement.

Solution de l'exercice 8.6

On note maintenant X_1 le vecteur de $n_1 + 2$ composantes $a_1 + jh$, $j = 0, \ldots, n_1 + 1$, des abscisses incluant a_1 et b_1. Les vecteurs X_2^i de l'algorithme 8.7 sont inchangés. La matrice du Laplacien est modifiée pour prendre en compte les conditions limites de Neumann et de Fourier. On utilise la représentation par bloc 8.6. Les matrices T et D sont maintenant de dimension $n_1 + 2 \times n_1 + 2$, et la matrice T est différente de 8.5 sur la première et la dernière ligne

$$T = \frac{1}{h^2} \begin{pmatrix} 3 + hc_{th} & -1 & 0 & \ldots & \ldots & 0 \\ -1 & 4 & -1 & \ddots & \ddots & \vdots \\ 0 & \ddots & 4 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 4 & -1 \\ 0 & \ldots & \ldots & 0 & -1 & 3 \end{pmatrix}.$$

Comme la température extérieure est égale à 0 °C, il n'y a aucune contribution de la condition de Fourier au second membre. L'algorithme de Schwarz est maintenant :

$$
\begin{array}{l}
\text{initialisation :} \\
\quad U_{a_2 g} = f_2(X_1), \quad U_{b_2 d} = g_2(X_1) \\
\quad U^{i,0}_{.,r} = (iU_{b_2 d} + (s-i)U_{a_2 g})/s, \quad \text{pour} \quad i = 2, \ldots, s \\
\quad B^i_{.,.} = F(X_1, X_2^i), \\
\text{pour} \quad p = 1, 2, \ldots, \quad \text{faire} \\
\quad \text{pour} \quad i = 1, \ldots, s \quad \text{faire} \\
\qquad \text{si } i = 1, \quad U^i_{a_2} = U_{a_2 g} \quad \text{sinon} \quad U^i_{a_2} = U^{i-1,p}_{.,n_2+1-r} \\
\qquad \text{si } i = s, \quad U^i_{b_2} = U_{b_2 d} \quad \text{sinon} \quad U^i_{b_2} = U^{i+1,p-1}_{.,r} \\
\qquad B^{i,p} = B^i, \quad B^{i,p}_{.,1} = B^{i,p}_{.,1} + U^i_{a_2}/h^2 \quad B^{i,p}_{.,n_2} = B^{i,p}_{.,n_2} + U^i_{b_2}/h^2 \\
\qquad \text{résoudre } AU^{i,p} = B^{i,p} \\
\qquad \text{si } i > 1, \quad R^i_{.,j} = U^{i,p}_{.,j} - U^{i-1,p}_{.,n2-r+1+j}, \quad \text{pour } j = 1, \ldots, r-1 \\
\quad \text{fin de} \quad i \\
\quad E^p = \sup_{i=2,\ldots,s} \|R^i\| \\
\text{si } E^p < \varepsilon \quad \text{fin de} \quad p
\end{array}
\tag{8.9}
$$

Cet algorithme est programmé dans `Schwarz2dFourier.m` et testé dans le troisième exemple du script `TestSchwarz2d`.

BIBLIOGRAPHIE

[Quarteroni et Valli, 1999] A. QUARTERONI, A. VALLI *Domain decomposition methods for partial differential equations*. Numerical Mathematics and Scientific Computation. The Clarendon Press Oxford University Press, New York, 1999. Oxford Science Publications.

Cet ouvrage permettra d'aborder en particulier la méthode de décomposition de domaines sans recouvrement, alternative de celle mise en œuvre dans ce projet.

Projet **9**

Modélisation géométrique : courbes et surfaces de Bézier

Fiche du projet

Difficulté :	2
Notions développées :	Courbes de Bézier, surfaces de Bézier
Domaines d'application :	Conception assistée par ordinateur (CAO), modélisation géométrique (CAGD)

La représentation des objets dans l'ordinateur est un besoin commun à de nombreuses disciplines comme la réalité virtuelle (RV), la conception assistée par ordinateur (CAO) ou encore la fabrication assistée par ordinateur (FAO). Dans tous les cas, la modélisation géométrique (CAGD pour Computer Aided Geometric Design) des objets est une étape indispensable. Elle consiste à donner une description de ces objets sous une forme mathématique autorisant les opérations (au sens ensembliste) auxquelles ils sont soumis (réunion, intersection, complément), afin de permettre la représentation graphique d'un élément simple (courbe, surface ou volume) ou composite, obtenu par composition de ces éléments simples.

Les premiers travaux dans ce domaine proviennent des milieux industriels et remontent aux années 1960. J. Ferguson (Boeing) et S. Coons (Ford) aux USA, P. de Casteljau (Citroën) et P. Bézier (Renault) en France, furent les pionniers de cette discipline. Dans ce chapitre, on propose une introduction à la problématique générale de la modélisation géométrique en étudiant quelques propriétés des *courbes de Bézier.*

9.1 LES COURBES DE BÉZIER

Considérons $m+1$ points de $\mathbb{R}^n$: $P_0, P_1, \ldots, P_m$, non nécessairement distincts, et un paramètre réel $t \in [0,1]$. On définit la *courbe de Bézier* $\mathcal{B}_m$ comme l'ensemble des points $P(t)$ obtenus par la formule (9.1) quand le paramètre t varie de 0 à 1 :

$$P(t) = \sum_{k=0}^{m} C_m^k t^k (1-t)^{m-k} P_k. \tag{9.1}$$

Les points $P_0, P_1, \ldots, P_m$ sont appelés les *points de contrôle* de la courbe $\mathcal{B}_m$. Les polynômes $B_m^k(t) = C_m^k t^k (1-t)^{m-k}$ sont les *polynômes de Bernstein* de degré m, qui vérifient les propriétés élémentaires suivantes :

$$\begin{cases} \forall t \in [0,1],\ 0 < B_m^k(t) < 1,\ \sum_{k=0}^{m} B_m^k(t) = 1. \\ B_m^k(0) = 0, \qquad \text{pour } 0 < k \leqslant m,\ B_m^0(0) = 1, \\ B_m^k(1) = 0, \qquad \text{pour } 0 \leqslant k < m,\ B_m^m(1) = 1. \end{cases} \tag{9.2}$$

On déduit de (9.2) que $P(0) = P_0$ et $P(1) = P_m$. Par contre rien ne permet d'indiquer que pour $0 < k < m$, le point P_k appartient à la courbe $\mathcal{B}_m$. En général, P_0 et P_m sont les seuls points de contrôle appartenant à la courbe. La figure 9.1 montre une courbe de Bézier définies dans $\mathbb{R}^2$ avec 5 points de contrôle. L'ordre des points de contrôle joue un rôle important dans la forme de la courbe, comme le montre la figure 9.2 dans laquelle la courbe de Bézier est définie par les mêmes points de contrôle que celle de la figure 9.1, mais avec échange des points extrêmes P_0 et P_4.

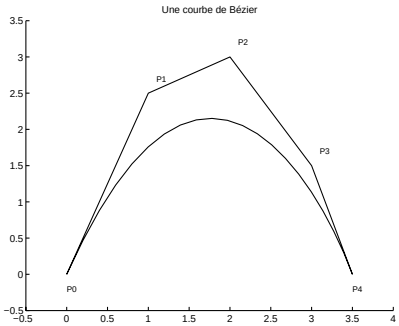


Figure 9.1 Une courbe de Bézier dans $\mathbb{R}^2$.

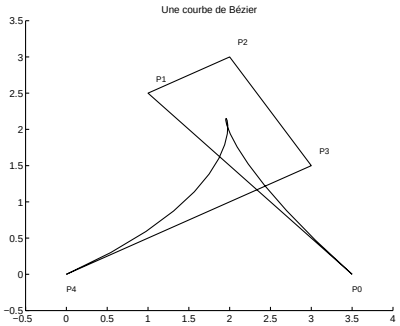


Figure 9.2 Une autre courbe de Bézier dans $\mathbb{R}^2$.

Cette définition permet donc de représenter de manière exacte et sous forme condensée des courbes du plan ($n = 2$) ou de l'espace ($n = 3$). Noter encore qu'à l'exception des extrêmités, une courbe de Bézier ne passe généralement pas par ses points de contrôle. Il est possible de définir des courbes qui passent effectivement par les points de contrôle, on les appelle *courbes d'interpolation.* Les courbes de Bézier définies selon (9.1) ne sont pas des courbes d'interpolation, mais font partie de la famille des *courbes splines.* Les courbes splines sont obtenues par la définition générale (9.3), dans laquelle les fonctions f_k sont des polynômes.

$$P(t) = \sum_{k=0}^{m} f_k(t) P_k. \tag{9.3}$$

On peut définir de cette manière une grande variété de courbes (B-splines, NURBS), suivant le choix des fonctions f_k (polynômes de degré $p \neq m$ ou fonctions rationnelles, etc.). Toutes ces courbes sont entièrement déterminées par la donnée des points de contrôle et le choix des fonctions f_k associées. Enfin on peut encore définir une courbe "par morceaux", comme réunion de courbes définies chacune selon la formule 9.3. Dans cette présentation, nous nous contenterons d'étudier les courbes de Bézier de degré m, définies par $m + 1$ points de contrôle.

9.2 PROPRIÉTÉS DES COURBES DE BÉZIER

Etudions maintenant quelques propriétés des courbes de Bézier dont certaines, d'un intérêt très pratique, permettent de comprendre le succès de ce type de représentation auprès des ingénieurs.

9.2.1 Enveloppe convexe des points de contrôle

Le point $P(t)$ est défini en (9.1) comme le barycentre des $m + 1$ points de contrôle P_k, affectés des poids $B_m^k(t)$. On déduit de la première propriété (9.2) que $P(t)$ appartient à l'enveloppe convexe des points P_k. La figure 9.3 montre une courbe de Bézier définie dans $\mathbb{R}^2$ contenue dans l'enveloppe convexe de ses points de contrôle. Cette enveloppe convexe contient en particulier le polygone $P_0P_1 \ldots P_mP_0$, appelé *polygone de contrôle.* Attention : le polygone de contrôle n'est pas convexe en général (comparer les figures 9.3 et 9.4).

9.2.2 Points de contrôle multiples

Dans la définition d'une courbe de Bézier, il n'est pas nécessaire de prendre les points de contrôle distincts. Cette souplesse permet de générer des courbes de formes assez complexes, fermées ou non, comme le montrent les figures 9.5 à 9.8.

9.2.3 Vecteur tangent à la courbe

Le vecteur tangent à la courbe de Bézier $\mathcal{B}_m$ au point $P(t)$ est défini par

$$\vec{\tau}(t) = \frac{d}{dt}P(t) = \sum_{k=0}^{m} \frac{d}{dt}B_m^k(t)P_k. \tag{9.4}$$

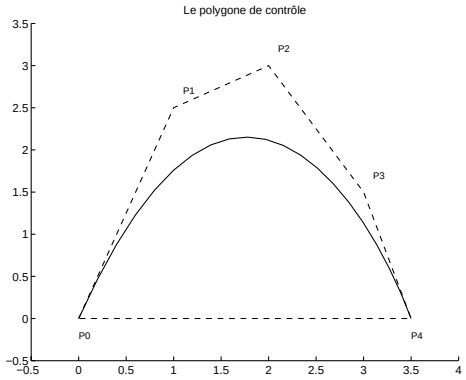


Figure 9.3 Un polygone de contrôle convexe.

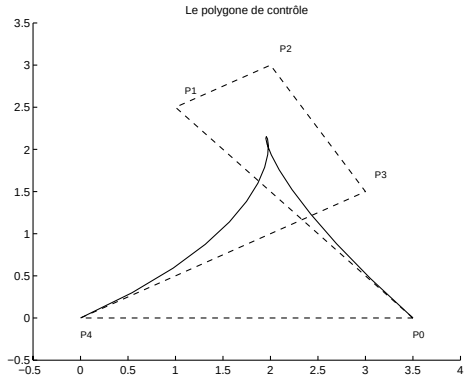


Figure 9.4 Un polygone de contrôle non convexe.

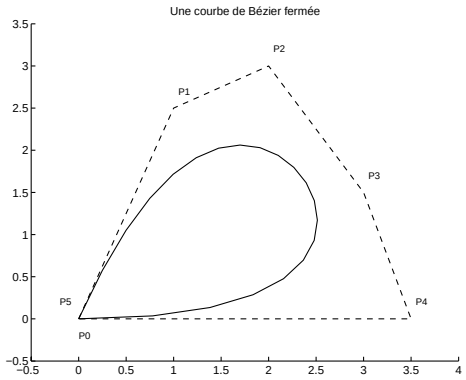


Figure 9.5 Utilisation des points multiples (1).

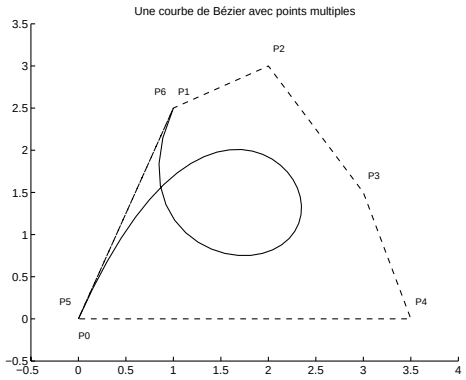


Figure 9.6 Utilisation des points multiples (2).

Calculons

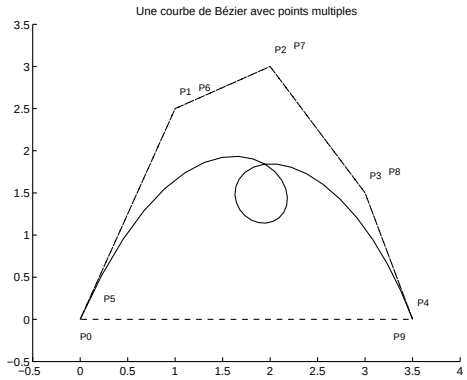
$$\frac{d}{dt}B_m^k(t) = \begin{cases} -m(1-t)^{m-1}, \text{ si } k = 0, \\ m(1-mt)(1-t)^{m-2}, \text{ si } k = 1, \\ C_m^k(k-mt)t^{k-1}(1-t)^{m-k-1}, \text{ si } 1 < k < m-1, \\ mt^{m-2}(m-1-mt), \text{ si } k = m-1, \\ mt^{m-1}, \text{ si } k = m. \end{cases}$$

On en déduit que si $P_0 \neq P_1$, alors le vecteur tangent en P_0 est $\vec{\tau}(0) = m\overrightarrow{P_0P_1}$. La courbe de Bézier $\mathcal{B}_m$ est tangente en P_0 au côté P_0P_1 du polygone de contrôle. De même, si $P_{m-1} \neq P_m$, alors $\vec{\tau}(1) = m\overrightarrow{P_{m-1}P_m}$: la courbe est tangente en P_m au côté $P_{m-1}P_m$. Cette propriété est visible sur les figures précédentes.

Remarque 9.1 : On montre encore la relation générale

$$\frac{d}{dt}B_m^k(t) = m(B_{m-1}^{k-1}(t) - B_{m-1}^k(t)).$$

qui peut être utilisée pour définir un algorithme de calcul du vecteur tangent.



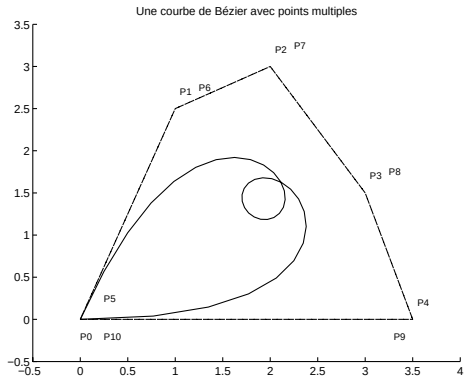


Figure 9.7 Utilisation des points multiples (3). **Figure 9.8** Utilisation des points multiples (4).

9.2.4 Raccord de deux courbes de Bézier

Soit $\mathcal{B}_m$ la courbe de Bézier définie par les $m+1$ points de contrôle $P_0, P_1, \ldots, P_m$ et $\mathcal{B}'_{m'}$ la courbe de Bézier définie par les $m'+1$ points de contrôle $P'_0, P'_1, \ldots, P'_{m'}$. On s'intéresse maintenant au problème de raccordement de ces deux courbes et en particulier à l'aspect visuel de ce raccord lorsque l'on trace les courbes sur un graphique. Il s'agit là d'un point important en CAO, où l'on recherche une qualité maximale des images.

Pour obtenir le raccord des courbes, il faut d'abord imposer la continuité $\mathcal{C}^0$. On suppose donc que $P_m = P'_0$. Ensuite, on sait que la courbe $\mathcal{B}_m$ est tangente en P_m au vecteur $P_{m-1}P_m$ et que la courbe $\mathcal{B}'_{m'}$ est tangente en P'_0 au vecteur $P'_0P'_1$. Le raccord tangentiel des deux courbes est donc obtenu à la condition que les points P_{m-1}, $P_m = P'_0$ et P'_1 soient alignés. En général cette condition est suffisante pour l'aspect esthétique du tracé. On parle alors de raccord $\mathcal{G}^1$, pour le distinguer du raccord $\mathcal{C}^1$, obtenu quand le vecteur tangent à la courbe $\vec{\tau}$ est continu au passage d'une courbe à l'autre. Sachant que pour la courbe $\mathcal{B}_m$, on a $\vec{\tau}(1) = m\overrightarrow{P_{m-1}P_m}$, tandis que pour la courbe $\mathcal{B}'_m$ $\vec{\tau}(0) = m\overrightarrow{P'_0P'_1}$, le raccord $\mathcal{C}^1$ est obtenu à la condition que

$\overrightarrow{P_{m-1}P_m} = \overrightarrow{P'_0P'_1}$. Le point $P_m = P'_0$ doit être le milieu du segment $P_{m-1}P'_1$. La figure 9.9 présente un raccord $\mathcal{G}^1$ tandis que la figure 9.10 montre un raccord $\mathcal{C}^1$. La différence entre les deux raccords n'est pas décelable sur les courbes, mais on voit sur la figure 9.10 que le point $P_4 \equiv P'_0$ est milieu du segment $P_3P'_1$, ce qui n'est pas le cas sur la figure 9.9. La différence entre ces types de raccords prend de l'importance lorsqu'il faut générer des points *uniformément* répartis sur la courbe $\mathcal{B} = \mathcal{B}_m \cup \mathcal{B}_m$.

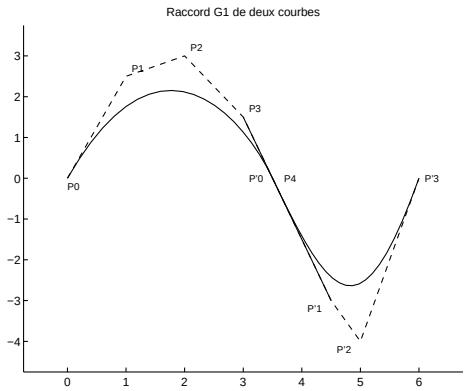


Figure 9.9 Raccord $\mathcal{G}^1$ de deux courbes.

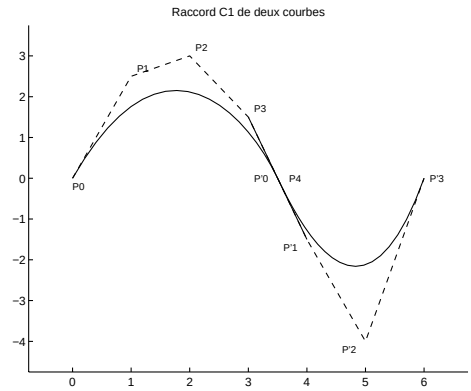


Figure 9.10 Raccord $\mathcal{C}^1$ de deux courbes.

9.2.5 Construction d'un point $P(t)$

Si la formule (9.1) définit mathématiquement un point d'une courbe de Bézier, la construction effective du point $P(t)$ correspondant à une valeur donnée $t \in [0,1]$ du paramètre selon (9.1) est extrêmement coûteuse en temps calcul. De plus l'évaluation de la valeur de polynômes de degré élevé étant sujette à des instabilités numériques, l'application de cette formule peut conduire à des résultats peu fiables. Heureusement, il est possible de construire le point $P(t)$ d'une manière stable et économique. On utilise pour cela une autre propriété des polynômes de Bernstein :

$$B^k_{m+1}(t) = tB^{k-1}_m(t) + (1-t)B^k_m(t). \tag{9.5}$$

Ce résultat est établi en écrivant simplement la relation

$$\begin{aligned} t\,B^{k-1}_m(t) + (1-t)B^k_m(t) &= t\,C^{k-1}_m t^{k-1}(1-t)^{m-k+1} + (1-t)C^k_m t^k(1-t)^{m-k} \\ &= (C^{k-1}_m + C^k_m)t^k(1-t)^{m-k+1} \\ &= C^k_{m+1}t^k(1-t)^{m+1-k} = B^k_{m+1}(t). \end{aligned}$$

Cette propriété est utilisée dans la pratique pour la construction des courbes de Bézier de la façon suivante : la valeur du paramètre $t \in [0,1]$ étant fixée, on définit

pour $p = 0, 1, \ldots, m$ et $q = p, p+1, \ldots, m$, les points P_q^p par les relations

$$\begin{array}{l}\text{pour } p = 0, \text{ initialisation :}\\ \quad \text{pour } q = 0, 1, \ldots, m, \text{ faire}\\ \quad\quad P_q^0 = P_q\\ \quad \text{fin}\\ \text{fin}\\ \text{pour } p = 1, 2, \ldots, m, \text{ faire}\\ \quad \text{pour } q = p, p+1, \ldots, m, \text{ faire}\\ \quad\quad P_q^p = tP_{q-1}^{p-1} + (1-t)P_q^{p-1}.\\ \quad \text{fin}\\ \text{fin}\end{array} \tag{9.6}$$

À l'étape p de l'algorithme (9.6), chacun des $m - p$ points P_q^p est défini comme barycentre des points P_{q-1}^{p-1} et P_q^{p-1} obtenus à l'étape précédente. Montrons par récurrence que le point P_q^p vérifie, pour $p = 0, 1, \ldots, m$ et $q = p, p+1, \ldots, m$, la relation

$$P_q^p = \sum_{j=0}^{m} B_p^j P_j.$$

La relation est vraie pour $p = 0$ par définition des points P_q^0. Supposons la vraie jusqu'à l'ordre $p-1$ inclus. Pour $q = p, p+1, \ldots, m$, on écrit que

$$P_q^p = tP_{q-1}^{p-1} + (1-t)P_q^{p-1} = \sum_{j=0}^{m} (tB_{q-1}^j + (1-t)B_q^j)P_j = \sum_{j=0}^{m} B_q^j P_j.$$

La relation est donc vraie pour tout p. En particulier pour $p = m$, on obtient

$$P_m^m = \sum_{j=0}^{m} B_m^j P_j = P(t).$$

Pour la valeur $t \in [0, 1]$ du paramètre, la construction du point $P(t)$ de la courbe de Bézier $\mathcal{B}_m$ de points de contrôle $P_0, P_1, \ldots, P_m$ est donc effectuée selon l'algorithme (9.6), appelé *algorithme de Casteljau.* Le coût calcul de $P(t)$ selon (9.6) est équivalent à celui de $m + \cdots + 1 = m(m+1)/2$ points. Cette construction est visualisée sur la figure 9.11.

9.3 CONSTRUCTION DE COURBES DE BÉZIER : MISE EN ŒUVRE

Il est temps de traiter quelques exemples pour mieux comprendre la facilité d'utilisation des courbes de Bézier. On va donc construire quelques courbes de Bézier en utilisant l'algorithme de Casteljau.

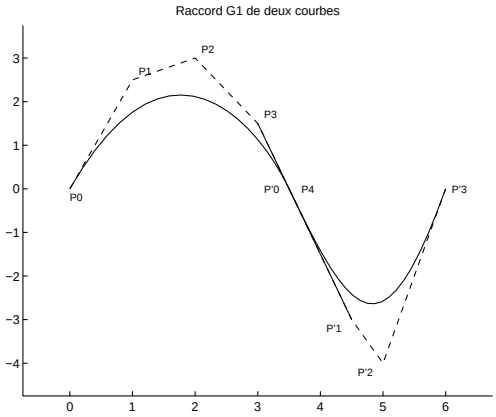


Figure 9.11 Construction du point $P(t)$.

Remarque 9.2 : Pour un premier temps on pourra dans l'exercice qui suit, construire la courbe de la figure 9.1. Pour cette courbe, $m = 4$ les points de contrôle sont $P_0 = (0, 0), P_1 = (1, 2.5), P_2 = (2, 3), P_3 = (3, 1.5), P_4 = (3.5, 0)$.

Exercice 9.1

1. Choisir $m + 1$ points $P_0, P_1, \ldots, P_m$ de $\mathbb{R}^2$.
2. Pour $t \in [0, 1]$, valeur fixée du paramètre, écrire une procédure de construction du point $P(t)$ de la courbe de Bézier de points de contrôle $P_0, P_1, \ldots, P_m$, en utilisant l'algorithme de Casteljau (9.6).
3. Construire et représenter graphiquement la courbe de Bézier de points de contrôle $P_0, P_1, \ldots, P_m$.
4. Répéter l'expérience en faisant varier le nombre et la répartition des points de contrôle.
5. Vérifier expérimentalement la condition de raccord.

La solution de cet exercice est présentée à la page 207.

Subdivision d'une courbe de Bézier

Soit $\mathcal{B}_m$ la courbe de Bézier définie par les $m + 1$ points de contrôle $P_0, P_1, \ldots, P_m$. Un point P de cette courbe est défini par la formule

$$P(t) = \sum_{k=0}^{m} C_m^k t^k (1-t)^{m-k} P_k \tag{9.7}$$

dans laquelle le paramètre t appartient à l'intervalle $[0, 1]$. Soit θ une valeur fixée dans l'intervalle $[0, 1]$, on lui associe par (9.7) le point $P(\theta)$ de la courbe $\mathcal{B}_m$. Pour

construire ce point selon l'algorithme de Casteljau (9.6), on construit successivement les points P_q^p, pour $p = 0, 1, \ldots, m$ et $q = p, p+1, \ldots, m$, et on obtient finalement $P(\theta) = P_m^m$.

Considérons maintenant la courbe de Bézier $\tilde{\mathcal{B}}_m$ définie par les $m+1$ points de contrôle $P_0^0, P_1^1, \ldots, P_m^m$. Un point $\tilde{P}$ de cette courbe est défini par la formule

$$\tilde{P}(\tilde{t}) = \sum_{k=0}^{m} C_m^k \tilde{t}^k (1-\tilde{t})^{m-k} P_k^k \tag{9.8}$$

dans laquelle le paramètre $\tilde{t}$ appartient à l'intervalle $[0, 1]$.

On va montrer que $\tilde{\mathcal{B}}_m$ est la portion de la courbe $\mathcal{B}_m$ obtenue quand le paramètre t varie dans l'intervalle $[0, \theta]$. Pour établir ce résultat, on montre d'abord que les points P_q^p générés par l'algorithme (9.6) vérifient, pour $1 \leqslant p \leqslant m$ et $p \leqslant q \leqslant m$, la relation

$$P_q^p = \sum_{k=0}^{p} C_p^k \theta^k (1-\theta)^{p-k} P_{q-k}. \tag{9.9}$$

Cette relation est établie par récurrence. Pour $p = 1$ et $1 \leqslant q \leqslant m$, on a par définition des points P_q^1 selon (9.6)

$$P_q^1 = \theta P_{q-1}^0 + (1-\theta) P_q^0 = \sum_{k=0}^{1} C_1^k \theta^k (1-\theta)^{1-k} P_{q-k}.$$

Supposons la propriété établie jusqu'à l'ordre $p-1$ inclus, alors pour $p \leqslant q \leqslant m$, on écrit

$$\begin{aligned} P_q^p &= \theta P_{q-1}^{p-1} + (1-\theta) P_q^{p-1} \\ &= \textstyle\sum_{k=0}^{p-1} C_{p-1}^k \theta^{k+1} (1-\theta)^{p-1-k} P_{q-1-k} + \sum_{k=0}^{p-1} C_{p-1}^k \theta^k (1-\theta)^{p-k} P_{q-k} \\ &= \textstyle\sum_{k=0}^{p-2} C_{p-1}^{p-1-k} \theta^{k+1} (1-\theta)^{p-1-k} P_{q-1-k} + \theta^p P_{q-p} \\ &\quad + (1-\theta)^p P_q + \textstyle\sum_{k=1}^{p-1} C_{p-1}^k \theta^k (1-\theta)^{p-k} P_{q-k} \end{aligned}$$

ainsi

$$\begin{aligned} P_q^p &= \textstyle\sum_{k=1}^{p-1} C_{p-1}^{p-k} \theta^k (1-\theta)^{p-k} P_{q-k} + \theta^p P_{q-p} \\ &\quad + (1-\theta)^p P_q + \textstyle\sum_{k=1}^{p-1} C_{p-1}^k \theta^k (1-\theta)^{p-k} P_{q-k} \\ &= \theta^p P_{q-p} + \textstyle\sum_{k=1}^{p-1} (C_{p-1}^{p-1-(k-1)} + C_{p-1}^k) \theta^k (1-\theta)^{p-k} P_{q-k} + (1-\theta)^p P_q \end{aligned}$$

et finalement

$$P_q^p = \sum_{k=0}^{p} C_p^k \theta^k (1-\theta)^{p-k} P_{q-k}.$$

Le point $\tilde{P}(\tilde{t})$ de la courbe de Bézier $\tilde{\mathcal{B}}_m$ définie par les $m+1$ points de contrôle $P_0^0, P_1^1, \ldots, P_m^m$, vérifie donc

$$\tilde{P}(\tilde{t}) = \sum_{k=0}^{m} C_m^k \tilde{t}^k (1-\tilde{t})^{m-k} P_k^k = \sum_{k=0}^{m} C_m^k \tilde{t}^k (1-\tilde{t})^{m-k} \sum_{l=0}^{k} C_k^l \theta^l (1-\theta)^{k-l} P_{k-l} \quad (9.10)$$

En regroupant dans cette somme les termes associés à chacun des points P_p, on obtient

$$\tilde{P}(\tilde{t}) = \sum_{p=0}^{m} \Big[\sum_{l=0}^{m-p} C_m^{p+l} C_{p+l}^p \theta^{p+l} (1-\theta)^{m-p-l} \tilde{t}^p (1-\tilde{t})^l \Big] P_p.$$

Ensuite on remarque que $C_m^{p+l} C_{p+l}^p = C_m^p C_{m-p}^l$, puis on écrit

$$\theta^{p+l}(1-\theta)^{m-p-l}\tilde{t}^p(1-\tilde{t})^l = (\theta\tilde{t})^p[(\theta - \theta\tilde{t})^l (1-\theta)^{m-p-l}].$$

On en déduit que la formule (9.10) s'écrit encore

$$\begin{array}{rl} \tilde{P}(\tilde{t}) & = \sum_{p=0}^{m} C_m^p (\theta\tilde{t})^p \Big[\sum_{l=0}^{m-p} C_{m-p}^l (\theta - \theta\tilde{t})^l (1-\theta)^{m-p-l} \Big] P_p \\ & = \sum_{p=0}^{m} C_m^p (\theta\tilde{t})^p (1-\theta\tilde{t})^{m-p} P_p. \end{array} \quad (9.11)$$

Pour une valeur θ fixée dans dans $[0,1]$, le produit $\theta\tilde{t}$ varie dans $[0,\theta]$ lorsque le paramètre $\tilde{t}$ varie dans $[0,1]$, Le point $\tilde{P}(\tilde{t})$ défini par (9.10) ou (9.11) parcourt donc toute la portion de la courbe de Bézier comprise entre les points P_0 et $P(\theta)$ lorsque $\tilde{t}$ décrit $[0,1]$.

Comment décrire l'autre partie de la courbe ? En inversant l'ordre des points de contrôle et en changeant θ en $1-\theta$. En effet le point $P(\theta)$ de la courbe de Bézier de points de contrôle $P_0, P_1, \ldots, P_m$, est défini selon la formule

$$P(\theta) = \sum_{k=0}^{m} C_m^k \theta^k (1-\theta)^{m-k} P_k = \sum_{k=0}^{m} C_m^k (1-\theta)^k \theta^{m-k} P_{m-k}.$$

Le point $P(\theta)$ est donc confondu avec le point $Q(1-\theta)$ de la courbe de Bézier de points de contrôle $P_m, P_{m-1}, \ldots, P_0$. Pour décrire la partie de la courbe comprise entre les points $P(\theta)$ et P_m, on construit selon l'algorithme (9.6) les points $Q_0^0, Q_1^1, \ldots, Q_m^m$ associés à la courbe de Bézier de points de contrôle $P_m, P_{m-1}, \ldots, P_0$, puis on pose

$$\tilde{Q}(t) = \sum_{k=0}^{m} C_m^k t^k (1-t)^{m-k} Q_k^k.$$

Remarque 9.3 : Ce résultat est encore vrai pour les courbes B-splines pour lesquelles on peut démontrer d'autres propriétés concernant le déplacement, la suppression ou l'insertion d'un point de contrôle.

9.4 INTERSECTION DE DEUX COURBES DE BÉZIER

On s'intéresse maintenant à l'étude de l'intersection de deux courbes de Bézier $\mathcal{B}_m$ et $\mathcal{B}'_{m'}$ définies dans $\mathbb{R}^2$ par leurs points de contrôle. On pose alors

$$P(t) = \sum_{k=0}^{m} C_m^k t^k (1-t)^{m-k} P_k \quad \text{et} \quad P'(t') = \sum_{k'=0}^{m'} C_{m'}^{k'} (t')^{k'} (1-t')^{m'-k'} P'_{k'}. \quad (9.12)$$

Remarque 9.4 : L'utilisation des notations P', t', etc. permet de simplifier l'écriture des formules qui suivent. Il ne s'agit pas de dérivation !

Existe-t-il des valeurs t et t' telles que $P(t) = P'(t')$? Comment les calculer ? D'après un résultat de géométrie algébrique, il est possible d'obtenir à partir de (9.12) une représentation implicite $f(x,y)$ et $f'(x,y)$ des courbes $\mathcal{B}_m$ et $\mathcal{B}'_{m'}$. Mais la recherche d'un éventuel point d'intersection conduit, sous cette forme, à l'étude des racines d'un polynôme de degré $m + m'$. Cette solution est trop complexe et numériquement coûteuse. On se propose d'effectuer la recherche d'un point d'intersection de deux courbes de Bézier à l'aide d'un algorithme plus simple, utilisant leurs propriétés. Le principe de cette recherche est le suivant : puisqu'une courbe de Bézier $\mathcal{B}_m$ est contenue dans l'enveloppe convexe $\mathcal{E}_m$ de ses points de contrôle, deux courbes $\mathcal{B}_m$ et $\mathcal{B}'_{m'}$ ne se coupent pas si l'intersection de leur enveloppe convexe $\mathcal{E}_m \cap \mathcal{E}'_{m'}$ est vide. Si l'ensemble $\mathcal{E}_m \cap \mathcal{E}'_{m'}$ n'est pas vide, il est possible que les courbes $\mathcal{B}_m$ et $\mathcal{B}'_{m'}$ se coupent. Pour préciser la situation, on subdivise chaque courbe en deux sous-courbes et on teste les intersections des 4 enveloppes convexes correspondantes. Pour subdiviser la courbe $\mathcal{B}_m$ en deux sous-courbes $\mathcal{B}_m^1$ et $\mathcal{B}_m^2$, on utilise la propriété énoncée précédemment. La courbe $\mathcal{B}_m^1$ correspond à la portion de la courbe $\mathcal{B}_m$ décrite quand $t \in [0, 1/2]$, tandis que la $\mathcal{B}_m^2$ correspond à la portion de la courbe $\mathcal{B}_m$ décrite quand $t \in [1/2, 1]$. Les points de contrôle des courbes $\mathcal{B}_m^1$ et $\mathcal{B}_m^2$ sont définis par la formule (9.9). Ce procédé est réitéré tant qu'il existe une intersection non vide entre deux enveloppes convexes. L'algorithme associé s'écrit

```
initialisation :
    E_1 = E_m,  E'_1 = E'_{m'}
fin
tant qu'il existe E_k ∩ E'_{k'} ≠ ∅ , faire
   subdiviser E_k = E_{k_1} ∪ E_{k_2}
   subdiviser E'_{k'} = E'_{k'_1} ∪ E'_{k'_2}
   tester E_{k_i} ∩ E'_{k'_j}        (*)
   fin
fin
```
(9.13)

À chaque étape, les nouveaux points de contrôles sont définis à l'intérieur de l'enveloppe convexe de la portion de courbe précédente. La mesure (aire) du polygone convexe, contour de l'enveloppe $\mathcal{E}_k$, est donc strictement décroissante. Ainsi l'algorithme (9.13) converge au sens suivant : ou bien toutes les intersections sont vides,

auquel cas les courbes $\mathcal{B}_m$ et $\mathcal{B}'_{m'}$ ne se coupent pas, ou bien il existe au moins une intersection non vide dont la mesure tend vers zéro, c'est-à-dire que l'enveloppe $\mathcal{E}_k$ tend vers le point commun aux deux courbes. Noter que l'algorithme (9.13) peut aussi traiter le cas où les courbes $\mathcal{B}_m$ et $\mathcal{B}'_{m'}$ se coupent plusieurs fois.

Il reste à tester de manière automatique l'intersection des ensembles $\mathcal{E}_k$ et $\mathcal{E}'_{k'}$, et donc dans un premier temps savoir calculer l'enveloppe convexe de points de $\mathbb{R}^2$. Pour cela on commence par représenter chaque $\mathcal{E}_k$ (dont le contour est un polygone convexe) comme réunion de triangles du plan dont les sommets sont les points de contrôle. Il s'agit donc d'un problème de *maillage* de l'ensemble $\mathcal{E}_k$. Ensuite il faut tester l'intersection des triangles décrivant $\mathcal{E}_k$ et avec ceux décrivant $\mathcal{E}'_{k'}$, subdiviser puis remailler les sous-ensembles, etc. Bien que concrètement réalisable, ce procédé est trop fastidieux à mettre en œuvre ici, nous allons donc le simplifier de la manière suivante : chaque ensemble $\mathcal{E}_k$ est contenu dans un rectangle défini à partir des coordonnées des points de contrôles : $R_k = [xmin, xmax, ymin, ymax]_k$, et de mesure la plus petite possible. Dans l'algorithme (9.13), on remplace alors la ligne (*) par *tester* $R_{k_i} \cap R_{k'_j}$. On perd ainsi un peu d'efficacité au sens où les rectangles R_k étant plus grands que les enveloppes convexes $\mathcal{E}_k$ la convergence de l'algorithme est ralentie. En compensation, la mise en œuvre est beaucoup plus simple.

Critère d'arrêt : il est nécessaire de définir une limite aux itérations de l'algorithme (9.13). Un critère d'arrêt efficace consiste à se donner une mesure limite τ pour le rectangle $R = R_k \cap R_{k'}$ (c'est-à-dire une longueur et une largeur minimale). Si la longueur ou la largeur du rectangle courant est inférieure à cette valeur, il est assimilé à un segment de droite M_1M_2. On termine alors en définissant $S = \mathcal{B}_m \cap \mathcal{B}'_{m'}$ comme l'intersection des diagonales de R. Pour placer le point S sur la courbe de Bézier $\mathcal{B}_m$, il faut trouver la valeur correspondante du paramètre t, telle que

$$S = S(t) = \sum_{k=0}^{m} C_m^k t^k (1-t)^{m-k} P_k.$$

Pour cela on fait une approximation linéaire, cohérente avec la définition de S, et on écrit que

$$t \approx t(M_2)\frac{x(S) - x(M_1)}{x(M_2) - x(M_1)} + t(M_1)\frac{x(M_2) - x(S)}{x(M_2) - x(M_1)} \tag{9.14}$$

On procède de la même façon pour déterminer la valeur du paramètre t' telle que

$$S = S(t') \sum_{k'=0}^{m'} C_{m'}^{k'} (t')^{k'} (1-t')^{m'-k'} P'_{k'}.$$

On écrit donc

$$t' \approx t'(M'_2)\frac{x(S) - x(M'_1)}{x(M'_2) - x(M'_1)} + t'(M'_1)\frac{x(M'_2) - x(S)}{x(M'_2) - x(M'_1)}. \tag{9.15}$$

Remarque 9.5 : Si $x(M_2) = x(M_1)$ on modifie la formule (9.14) en utilisant les ordonnées : $y(M_2) - y(M_1)$ (même traitement éventuel pour les points M'_1 et M'_2).

Remarque 9.6 : On peut modifier le critère de convergence de l'algorithme en remplaçant le calcul du rectangle enveloppant la courbe par celui d'une mesure de l'écart de la courbe avec un segment de droite. Pour cela, on estime sa courbure (ou sa convexité) par une formule du type

$$h = \max_k(|x_{k-1} - 2x_k + x_{k+1}|, |y_{k-1} - 2y_k + y_{k+1}|).$$

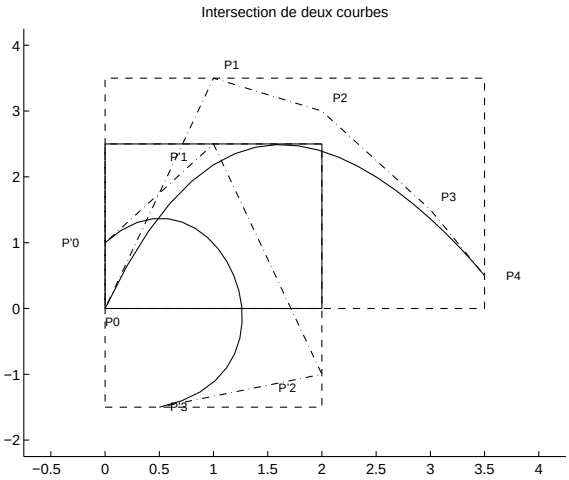


Figure 9.12 Intersection de deux courbes de Bézier.

Mise en œuvre

Exercice 9.2

1. Construire deux courbes de Bézier $\mathcal{B}_m$ et $\mathcal{B}'_{m'}$ définies dans $\mathbb{R}^2$ par leurs points de contrôle
2. Mettre en œuvre l'algorithme d'intersection (9.13) simplifié.
3. Calculer les paramètres t et t' selon (9.14), puis les points $P(t)$ et $P'(t')$. Comparer au résultat de (9.13).

La solution de cet exercice est présentée à la page 208.

9.5 SURFACES DE BÉZIER

Une représentation efficace des surfaces est obtenue de manière semblable en utilisant deux paramètres t_1 et t_2. Soient m_1 et m_2 deux entiers non nuls, et $(m_1+1)\times(m_2+1)$ points de contrôle $P_{k_1,k_2} \in \mathbb{R}^3$, on définit pour $(t_1, t_2) \in [0,1]\times[0,1]$ le point $P(t_1, t_2)$ par la relation

$$P(t_1,t_2) = \sum_{k_1=0}^{m_1}\sum_{k_2=0}^{m_2} C_{m_1}^{k_1} C_{m_2}^{k_2} t_1^{k_1}(1-t_1)^{m_1-k_1} t_2^{k_2}(1-t_2)^{m_2-k_2} P_{k_1,k_2}. \tag{9.16}$$

Quand (t_1, t_2) varie dans $[0,1]\times[0,1]$, le point $P(t_1, t_2)$ décrit une surface Bézier, appelée *carreau de Bézier* par les spécialistes de la CAO.

Remarque 9.7 : La définition théorique de la surface (9.16) n'impose aucune condition sur la disposition des points P_{k_1,k_2} dans l'espace. Cependant la répartition de ces points joue un rôle important sur l'aspect final de la surface générée. Pour obtenir une *belle* surface, ou pour la modifier *facilement* en déplaçant quelques points de contrôle, il est préférable de définir les points P_{k_1,k_2} suivant une *grille*.

Remarque 9.8 : Les carreaux ainsi définis sont dits *rectangulaires*, par opposition aux *carreaux triangulaires*, pour lesquels le point courant est défini par la relation

$$P(t_1,t_2,t_3) = \sum_{k_1+k_2+k_3=m} \frac{m!}{k_1!k_2!k_3!} t_1^{k_1} t_2^{k_2} t_3^{k_3} P_{k_1,k_2,k_3}. \tag{9.17}$$

Remarque 9.9 : On conçoit qu'avec la définition (9.17) la répartition des points de contrôle est plus délicate à gérer qu'avec la définition (9.16).

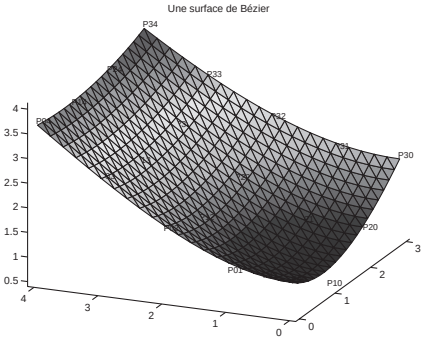


Figure 9.13 Une surface de Bézier et ses points de contrôle.

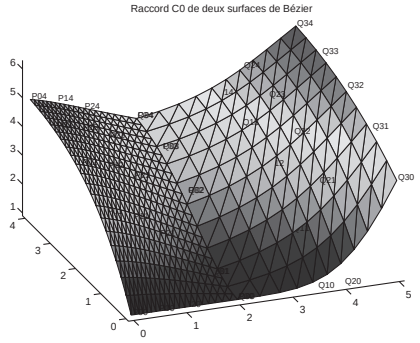


Figure 9.14 Raccord $\mathcal{C}^0$ de deux surfaces.

Propriétés des surfaces de Bézier

➤ Enveloppe convexe

Le point $P(t)$ est le barycentre des $(m_1+1)\times(m_2+1)$ points de contrôle P_{k_1,k_2} affectés des poids $B_{m_1}^{k_1}(t_1)B_{m_2}^{k_2}(t_2)$. Il découle encore de (9.2) que $P(t)$ appartient à l'enveloppe convexe des points P_{k_1,k_2}, c'est-à-dire le *polytope de contrôle* $P_{0,0}$, $P_{1,0}$, …, P_{m_1,m_2}.

➤ Vecteur tangent

Le plan tangent à la surface de Bézier au point $P(t_1,t_2)$ est défini par les deux vecteurs tangents $\vec{\tau}_1(t_1,t_2) = \frac{d}{dt_1}P(t_1,t_2)$ et $\vec{\tau}_2(t_1,t_2) = \frac{d}{dt_2}P(t_1,t_2)$. On en déduit que si $P_{0,0} \neq P_{1,0}$ et $P_{0,0} \neq P_{0,1}$, la surface de Bézier est tangente en $P_{0,0}$ au triangle $P_{1,0}P_{0,0}P_{0,1}$, facette du polytope de contrôle. On retrouve une propriété analogue aux sommets $P_{m_1,0}$, P_{0,m_2} et P_{m_1,m_2}.

➤ Raccord de surfaces de Bézier

Soit $\mathcal{S}_{m_1,m_2}$ la surface de Bézier définie par les $(m_1+1)\times(m_2+1)$ points de contrôle P_{k_1,k_2} $(1 \leqslant k_1 \leqslant m_1,\ 1 \leqslant k_2 \leqslant m_2)$ et $\mathcal{S}'_{m_1,m_2}$ la surface de Bézier définie par les $(m'_1+1)\times(m'_2+1)$ points de contrôle $P'_{k'_1,k'_2}$ $(1 \leqslant k'_1 \leqslant m'_1,\ 1 \leqslant k'_2 \leqslant m'_2)$. On s'intéresse au problème de raccordement de ces deux surfaces.

Le simple raccord des surfaces correspond à la continuité $\mathcal{C}^0$. Il doit donc exister une courbe commune aux bords des deux surfaces, ce qui revient à dire que les points de contrôle associés sont identiques. Supposons que $P_{m_1,k_2} = P_{0,k_2}$ pour $k_2 = 0,1,\ldots,m_2$. Les vecteurs $\vec{\tau}_2(1,t_2) = \frac{d}{dt_2}P(1,t_2)$ et $\vec{\tau}'_2(0,t'_2) = \frac{d}{dt'_2}P'(0,t'_2)$ sont donc égaux pour $t_2 \in [0,1]$. Pour assurer la continuité $\mathcal{G}^1$ il faut que pour $t_2 \in [0,1]$ les vecteurs $\vec{\tau}_1(1,t_2) = \frac{d}{dt_1}P(1,t_2)$ et $\vec{\tau}'_1(0,t'_2) = \frac{d}{dt'_1}P'(0,t'_2)$ soient colinéaires. Pour assurer la continuité $\mathcal{G}^1$, il faut que $\vec{\tau}_1(1,t_2) = \vec{\tau}'_1(0,t'_2)$ pour $t_2 \in [0,1]$. Cette dernière condition revient à dire que, pour $k_2 = 0,1,\ldots,m_2$, le point $P_{m_1,k_2} = P'_{0,k_2}$ est le milieu du segment $P_{m_1-1,k_2}P'_{1,k_2}$. La figure 9.14 présente un raccord $\mathcal{C}^0$. La figure 9.15 présente un raccord $\mathcal{G}^1$ tandis que la figure 9.16 montre un raccord $\mathcal{C}^1$. La différence entre les trois types de raccord est plus visible que dans le cas des courbes.

➤ Construction d'un point $P(t)$

Pour les carreaux de Bézier rectangulaires, la définition (9.16) par produit tensoriel permet de construire le point $P(t_1,t_2)$ par un algorithme analogue à (9.6). On écrit pour cela

$$P(t_1,t_2) = \sum_{k_1=0}^{m_1} C_{m_1}^{k_1} t_1^{k_1}(1-t_1)^{m_1-k_1} P_{k_1}, \quad \text{avec} \quad P_{k_1} = \sum_{k_2=0}^{m_2} C_{m_2}^{k_2} t_2^{k_2}(1-t_2)^{m_2-k_2} P_{k_1,k_2}.$$

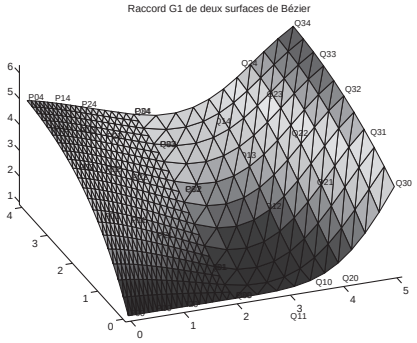


Figure 9.15 Raccord $\mathcal{G}^1$ de deux surfaces.

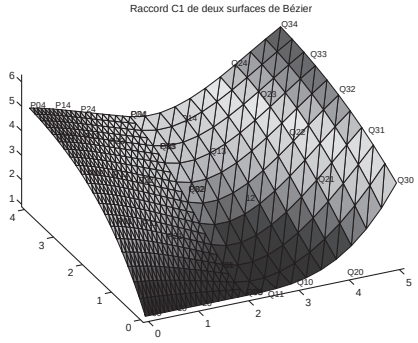


Figure 9.16 Raccord $\mathcal{C}^1$ de deux surfaces.

Pour k_1 fixé, le point P_{k_1} apparait comme un point $P_{k_1}(t_2)$ de la courbe de Bézier définie par les $m_2 + 1$ points de contrôle $P_{k_1,0}, P_{k_1,1}, \ldots, P_{k_1,m_2}$. Il peut donc être construit selon l'algorithme (9.6). Lorsque les $m_1 + 1$ points P_{k_1} sont déterminés, une nouvelle application de l'algorithme (9.6) permet de construire le point $P(t_1, t_2)$ de la surface de Bézier. Cette surface est ainsi définie par l'ensemble des facettes de sommets $P(t_1, t_2)$, $P(t_1 + \Delta t_1, t_2)$, $P(t_1 + \Delta t_1, t_2 + \Delta t_2)$ et $P(t_1, t_2 + \Delta t_2)$, avec Δt_1 et Δt_1 pas d'échantilonnage des variables t_1 et t_2. L'algorithme de construction du point $P(t_1, t_2)$ selon cette méthode est appelé *algorithme de De Boor - Coox.* Il s'écrit

pour $p_1 = 0$, initialisation :
  pour $q_1 = 0, 1, \ldots, m_1$, faire
    construction du point P_{q_1} selon (9.6)
      pour $p_2 = 0$, initialisation :
        pour $q_2 = 0, 1, \ldots, m_2$, faire
          $P^0_{q_1,q_2} = P_{q_1,q_2}$
        fin
      fin
      pour $p_2 = 1, 2, \ldots, m_2$, faire
        pour $q_2 = p_2, p_2 + 1, \ldots, m_2$, faire
          $P^{p_2}_{q_1,q_2} = t_2 P^{p_2-1}_{q_1,q_2-1} + (1 - t_2) P^{p_2-1}_{q_1,q_2}$. (9.18)
        fin
      fin : $P_{q_1} = P_{q_1,m_2}$
    fin de la construction du point P_{q_1}, poser : $P^0_{q_1} = P_{q_1}$
  fin
pour $p_1 = 1, 2, \ldots, m_1$, faire
  pour $q_1 = p_1, p_1 + 1, \ldots, m_1$, faire
    $P^{p_1}_{q_1} = t_1 P^{p_1-1}_{q_1-1} + (1 - t_1) P^{p_1-1}_{q_1}$.
  fin
fin : $P(t_1, t_2) = P^{m_1}_{m_1}$

9.6 CONSTRUCTION DE SURFACES DE BÉZIER : MISE EN ŒUVRE

Exercice 9.3

1. Choisir $(m_1 + 1) \times (m_2 + 1)$ points de contrôle $P_{k_1,k_2} \in \mathbb{R}^3$ dans $\mathbb{R}^3$.
2. Pour $(t_1, t_2) \in [0, 1] \times [0, 1]$, écrire une procédure de construction du point $P(t_1, t_2)$ en utilisant l'algorithme (9.18).
3. Construire et représenter la surface de Bézier de points de contrôle : $P_{0,0}, P_{1,0}, \ldots, P_{m_1,m_2}$.
4. Vérifier expérimentalement la condition de raccord.

La solution de cet exercice est présentée à la page 208

Remarque 9.10 : Pour les courageux : on peut aussi écrire une procédure de construction de l'intersection de deux surfaces de Bézier utilisant la procédure calculant l'intersection de deux courbes généralisée à $\mathbb{R}^3$ . . .

9.7 SOLUTIONS ET PROGRAMMES

Toutes les procédures nécessaires à la résolution des exercices de ce chapitre peuvent être téléchargées à partir de la page web du livre.

Solution de l'exercice 9.1

La procédure du fichier `caoexo1.m` définit un ensemble de points de contrôle et réalise la construction et la visualisation de la courbe de Bézier associée, représentée sur la figure 9.1. Cette procédure appelle la procédure du fichier `cbezier.m` qui calcule un échantillonnage de points à l'aide de la procédure du fichier `casteljau.m`, qui construit un point d'une courbe de Bézier selon l'algorithme de Casteljau (9.6).

Listing 9.1 *casteljau.m*

```
function [x,y]=casteljau(t,XP,YP)
%%
%% Construction d'un point d'une courbe de Bézier
%% selon l'algorithme de Casteljau
%%
m=size(XP,2)-1;
xx=XP;yy=YP;
for kk=1:m
xxx=xx; yyy=yy;
```

```
for k=kk:m
xx(k+1)=(1-t)*xxx(k)+t*xxx(k+1);
yy(k+1)=(1-t)*yyy(k)+t*yyy(k+1);
end
end
x=xx(m+1);y=yy(m+1);
```

La procédure du fichier `caoexo1.m` appelle ensuite la procédure du fichier `tbezier.m`, qui trace la courbe et les points de contrôle. Pour obtenir le tracé du polygone de contrôle, comme sur la la figure 9.1, on utilise la procédure du fichier `caoexo1b.m`, qui appelle `pbezier.m`. En échangeant les points P_1 et P_3, on obtient une courbe de Bézier différente, avec un polygone de contrôle non convexe (voir la procédure du fichier `caoexo1b.m`).

Les procédures `raccordCC0.m`, `raccordCG1.m` et `raccordCC1.m` permettent de visualiser le raccord C^0 (respectivement G^1 et C^1) de deux surfaces de Bézier. Ces deux derniers exemples correspondent aux figures 9.15 et 9.16.

Solution de l'exercice 9.2

La procédure du fichier `caoexo2.m` définit deux ensembles de points de contrôle et réalise la construction et la visualisation des deux courbes de Bézier associées. Chaque courbe est ensuite localisée dans un rectangle par la procédure du fichier `drectan.m`. L'intersection éventuelle de ces rectangles est testée par la procédure `rbezier.m`. En cas de succès, on lance la procédure du fichier `dbezier.m`, qui effectue une recherche itérative et adaptative d'un point d'intersection des deux courbes, selon l'algorithme 9.13. Lorsque le rectangle d'intersection a atteint une taille suffisamment petite, on calcule les coordonnées du point d'intersection, puis une approximation de la valeur correspondante du paramètre, selon les formules (9.14) et (9.15).

Cette procédure appelle la procédure du fichier `cbezier.m` qui construit un échantillonnage de points d'une courbe de Bézier à l'aide de la procédure du fichier `casteljau.m`, ainsi que la procédure du fichier `tbezier.m` qui trace la courbe et les points de contrôle.

Solution de l'exercice 9.3

Dans le fichier des solutions, la procédure du fichier `caoexo3.m` définit un ensemble de points de contrôle et réalise la construction et la visualisation de la surface de Bézier associée, représentée sur la figure 9.13. Cette procédure appelle la procédure du fichier `sbezier.m` qui construit un échantillonnage de points d'une surface de Bézier à l'aide de la procédure du fichier `coox.m`, ainsi que la procédure du fichier `ubezier.m` qui trace la surface et les points de contrôle. La procédure du fichier `coox.m` construit un point d'une surface de Bézier selon l'algorithme de De Boor - Coox (9.18).

Listing 9.2 *coox.m*

```
function [x,y,z]=coox(t1,t2,XP,YP,ZP)
%%
%% Construction d'un point d'une surface de Bézier
%% selon l'algorithme de De Boor - Coox
%%
np1=size(XP,1);np2=size(XP,2);
xx1=zeros(np1,1);yy1=zeros(np1,1);zz1=zeros(np1,1);
for k1=1:np1
xx2=zeros(np2,1);yy2=zeros(np2,1);zz2=zeros(np2,1);
for k2=1:np2
xx2(k2)=XP(k1,k2);yy2(k2)=YP(k1,k2);zz2(k2)=ZP(k1,k2);
end
[x,y,z]=cast3d(t2,xx2,yy2,zz2);
xx1(k1)=x;yy1(k1)=y;zz1(k1)=z;
end
[x,y,z]=cast3d(t1,xx1,yy1,zz1);
```

Dans le fichier des solutions, la procédure `raccordSC0.m` permet de visualiser le raccord C^0 de deux surfaces de Bézier. Les procédures `raccordSG1.m` et `raccordSC1.m` visualisent les raccords G^1 et C^1 de deux surfaces de Bézier. Ces trois exemples correspondent aux figures 9.14, 9.15 et 9.16.

BIBLIOGRAPHIE COMPLÉMENTAIRE OU LECTURE POUR APPROFONDIR

[Bezier-1986] P. Bézier *Courbes et surfaces*, Mathématiques et CAO (vol 4) HERMES. Paris (1986)

[Casteljau-1985] P. de Casteljau *Formes à pôles*, Mathématiques et CAO (vol 2) HERMES. Paris (1985)

[Farin-1992] G. Farin *Courbes et surfaces pour la CGAO*, Masson. Paris (1992)

[Risler-1991] J.-J. Risler Méthodes mathématiques pour la CAO Masson. Paris (1991)

[Piegl-Tiller-1995] L. Piegl et W. Tiller *The NURBS book*, Springer. Berlin (1995)

[Hoschek-Lasser-1997] J. Hoschek et D. Lasser *Fundamentals of Computer Aided Geometric Design*, Peters. Massachusetts (1997)

Projet **10**

Problème de Riemann et discontinuités. Étude du tube à choc

Fiche du projet

Difficulté :	3
Notions développées :	Système hyperbolique non-linéaire, équations d'Euler, schémas centrés : Lax-Wendroff, Mac-Cormack ; schémas décentrés : Godunov, Roe
Domaines d'application :	Tube à choc, écoulements supersoniques

10.1 PROBLÈME PHYSIQUE DU TUBE À CHOC

Le principe du tube à choc est le suivant : on considère un canal fermé à ses deux extrémités, contenant un gaz inerte qu'un diaphragme maintient dans des conditions thermodynamiques – pression, masse volumique et température – différentes (voir la figure 10.1). La rupture brutale du diaphragme provoque la formation immédiate d'une onde de discontinuité (onde de choc) qui se propage dans la section de basse pression. Ce dispositif, inventé en 1899 par Paul Vieille pour l'étude de déflagrations des charges explosives, constitue actuellement la soufflerie supersonique la plus éco-

nomique. Il sert à simuler les écoulements autour des projectiles, des avions supersoniques ou des navettes spatiales pendant la rentrée atmosphérique.

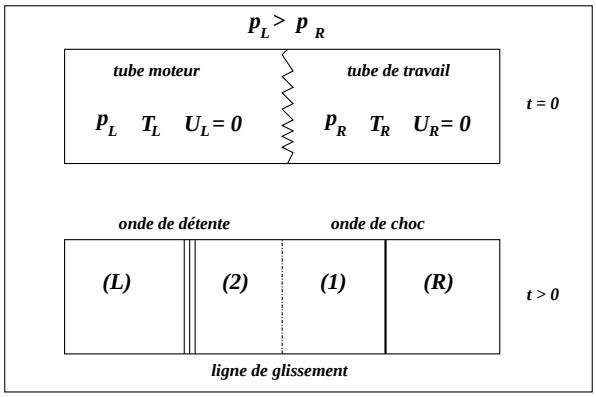


Figure 10.1 Représentation schématique du tube à choc à l'instant initial ($t = 0$) et des ondes qui se propagent dans le tube pour $t > 0$.

Considérons le tube à choc représenté schématiquement sur la figure 10.1. On appelle *tube moteur* la partie de plus grande pression ; dans notre cas, il s'agit de la partie gauche, caractérisée par la pression p_L, la masse volumique ρ_L, la température T_L et la vitesse initiale $U_L = 0$. La partie droite, décrite par les grandeurs $p_R < p_L$, ρ_R, T_R et $U_R = 0$, constitue le *tube de travail*. C'est dans cette partie qu'une éventuelle maquette d'avion sera placée.

A l'instant $t = 0$ le diaphragme est rompu ; un processus d'égalisation des pressions s'établit alors. Le gaz à haute pression se détend par une onde de détente et s'écoule dans le tube de travail. Il crée une onde de choc dans le gaz à basse pression qui est comprimé. Le gaz détendu est séparé du gaz comprimé par une ligne de glissement, qui peut être assimilée à une membrane fictive qui se déplace avec une vitesse constante (voir la figure 10.1).

Les ondes de détente, de choc et les lignes de glissement se caractérisent par des discontinuités de certaines fonctions décrivant l'écoulement ($p(x)$, ou $\rho(x)$, ou $T(x)$, ou $U(x)$).

10.2 SYSTÈME D'ÉQUATIONS D'EULER 1D

Pour décrire plus facilement les phénomènes physiques dans le tube à choc, on fait les hypothèses suivantes : le diaphragme s'ouvre instantanément et complètement ; le gaz est parfait ; on néglige les processus de diffusion moléculaire ; le tube est assez long pour ignorer les réflexions des ondes à ses extrémités.

Sous ces hypothèses, l'écoulement monodimensionnel (1D) qui s'établit dans le tube à choc est décrit par le système d'équations d'Euler [Bonnet et Luneau, 1989, Hirsh, 1988, LeVeque, 1992] :

$$\frac{\partial}{\partial t}\underbrace{\begin{pmatrix}\rho\\ \rho U\\ E\end{pmatrix}}_{W(x,t)}+\frac{\partial}{\partial x}\underbrace{\begin{pmatrix}\rho U\\ \rho U^2+p\\ (E+p)U\end{pmatrix}}_{F(W)}=0, \tag{10.1}$$

où ρ est la masse volumique du gaz donnée par la lois d'état du gaz parfait

$$p=\rho\mathcal{R}T, \tag{10.2}$$

et E l'énergie totale

$$E=\frac{p}{\gamma-1}+\frac{\rho}{2}U^2. \tag{10.3}$$

Les paramètres γ et $\mathcal{R}$ sont deux constantes caractérisant le gaz parfait. Il est également utile de définir localement la vitesse du son a, le nombre de Mach (M) et l'enthalpie totale (H) :

$$a=\sqrt{\gamma\mathcal{R}T}=\sqrt{\gamma\frac{p}{\rho}},\quad M=\frac{U}{a},\quad H=\frac{E+p}{\rho}=\frac{a^2}{\gamma-1}+\frac{1}{2}U^2. \tag{10.4}$$

Si les inconnues du problème sont groupées dans le vecteur $W=(\rho,\rho U,E)$, le système d'EDP à résoudre devient (noter que la pression n'est pas une inconnue car elle s'exprime en fonction des composantes de W par la formule 10.3) :

$$\frac{\partial W}{\partial t}+\frac{\partial}{\partial x}F(W)=0, \tag{10.5}$$

avec la condition initiale (on note par x_0 l'abscisse du diaphragme) :

$$W(x,0)=\begin{cases}(\rho_L,\rho_L U_L,E_L), & x\leqslant x_0\\ (\rho_R,\rho_R U_R,E_R), & x>x_0.\end{cases} \tag{10.6}$$

L'analyse mathématique du système d'EDP (10.5) considère sa forme *quasi-linéaire* qui s'écrit (noter l'analogie avec l'équation de convection présentée dans le Projet 1 de cet ouvrage) :

$$\frac{\partial W}{\partial t}+A\frac{\partial W}{\partial x}=0, \tag{10.7}$$

avec la matrice jacobienne

$$A=\frac{\partial F}{\partial W}=\begin{pmatrix}0 & 1 & 0\\ \frac{1}{2}(\gamma-3)U^2 & (3-\gamma)U & \gamma-1\\ \frac{1}{2}(\gamma-1)U^3-UH & H-(\gamma-1)U^2 & \gamma U\end{pmatrix}. \tag{10.8}$$

Il est intéressant d'observer que la matrice A vérifie la relation remarquable suivante :

$$AW=F(W). \tag{10.9}$$

De plus, on peut facilement calculer ses valeurs propres :

$$\lambda^0 = U, \quad \lambda^+ = U + a, \quad \lambda^- = U - a, \tag{10.10}$$

et les vecteurs propres correspondants :

$$v^0 = \begin{pmatrix} 1 \\ U \\ \frac{1}{2}U^2 \end{pmatrix}, \quad v^+ = \begin{pmatrix} 1 \\ U + a \\ H + aU \end{pmatrix}, \qquad v^- = \begin{pmatrix} 1 \\ U - a \\ H - aU \end{pmatrix}. \tag{10.11}$$

La matrice jacobienne A étant diagonalisable, avec des valeurs propres réelles, le système (10.7) est dit *hyperbolique*. Plus exactement, on peut décomposer A sous la forme :

$$A = P\Lambda P^{-1}, \Lambda = \begin{pmatrix} U-a & 0 & 0 \\ 0 & U & 0 \\ 0 & 0 & U+a \end{pmatrix}, P = \begin{pmatrix} 1 & 1 & 1 \\ U-a & U & U+a \\ H-aU & \frac{1}{2}U^2 & H+aU \end{pmatrix}. \tag{10.12}$$

On vérifiera que

$$P^{-1} = \begin{pmatrix} \frac{1}{2}\left(\alpha_1 + \frac{U}{a}\right) & -\frac{1}{2}\left(\alpha_2 U + \frac{1}{a}\right) & \frac{\alpha_2}{2} \\ 1 - \alpha_1 & \alpha_2 U & -\alpha_2 \\ \frac{1}{2}\left(\alpha_1 - \frac{U}{a}\right) & -\frac{1}{2}\left(\alpha_2 U - \frac{1}{a}\right) & \frac{\alpha_2}{2} \end{pmatrix}, \quad \alpha_1 = \frac{\gamma - 1}{2}\frac{U^2}{a^2}, \quad \alpha_2 = \frac{\gamma - 1}{a^2}. \tag{10.13}$$

Une propriété intéressante des systèmes hyperboliques (pour une analyse plus rigoureuse des systèmes hyperboliques, voir [Godlewski et Raviart, 1996]) est l'existence des courbes *caractéristiques* le long desquelles des variables particulières, appelées *invariants*, sont constantes. Cette manière particulière de propagation de l'information est très importante d'un point de vue pratique, car elle permet de calculer la solution en tout point P du plan (x, t) en réunissant les informations (invariants) apportées au point P par les caractéristiques venant de régions où la solution est déjà calculée.

De manière générale, l'équation décrivant une caractéristique est $dx/dt = \lambda$, avec λ valeur propre de la matrice jacobienne. L'invariant r correspondant étant constant le long de la caractéristique, il doit vérifier :

$$\frac{dr}{dt} = \frac{\partial r}{\partial t} + \frac{\partial r}{\partial x}\frac{dx}{dt} = 0, \quad \text{donc} \quad \frac{\partial r}{\partial t} + \lambda\frac{\partial r}{\partial x} = 0. \tag{10.14}$$

Pour l'équation de convection (voir le Projet 1) $\partial u/\partial t + c\,\partial u/\partial x = 0$, avec $c = const$, la seule caractéristique est la droite $x = c\,t$ et l'invariant est la solution même, $r = u$. Dans le cas du système (10.7), il existe trois caractéristiques distinctes :

$$C^0 : \frac{dx}{dt} = U, \quad C^+ : \frac{dx}{dt} = U + a, \quad C^- : \frac{dx}{dt} = U - a, \tag{10.15}$$

avec les invariants correspondants (dans le cas d'un fluide d'entropie constante, ou *isentropique*[1]) :

$$r^0 = p/\rho^\gamma, \quad r^+ = U + \frac{2a}{\gamma - 1}, \quad r^- = U - \frac{2a}{\gamma - 1}. \tag{10.16}$$

Cette analyse en caractéristiques sera utilisée pour le calcul de la solution exacte du tube à choc.

Définition 10.1 *Le système d'EDP (10.5) – non-linéaire, hyperbolique – avec la condition initiale (10.6) – discontinue et constante par morceaux – constitue un problème de Riemann.*

10.2.1 Adimensionnement des équations

Pour les applications numériques, il est convenable de travailler avec des grandeurs sans dimension (les erreurs d'arrondis sont réduites). Par conséquent, les grandeurs physiques intervenant dans les équations précédentes seront écrites sous la forme (l'état de référence est choisi en fonction des paramètres du tube de travail) :

$$\rho = \rho^* \cdot \rho_R, \quad U = U^* \cdot a_R, \quad a = a^* \cdot a_R, \quad T = T^* \cdot (\gamma T_R), \tag{10.17}$$
$$t = t^* \cdot t_0, \quad x = x^* \cdot (a_R t_0), \quad p = p^* \cdot (\rho_R a_R^2) = p^* \cdot (\gamma p_R), \quad E = E^* \cdot (\rho_R a_R^2).$$

Les équations pour les variables sans dimension s'écrivent sous la même forme que précédemment :

$$\frac{\partial}{\partial t^*} \underbrace{\begin{pmatrix} \rho^* \\ \rho^* U^* \\ E^* \end{pmatrix}}_{W^*(x^*,t^*)} + \frac{\partial}{\partial x^*} \underbrace{\begin{pmatrix} \rho^* U^* \\ \rho^* U^{*2} + p^* \\ (E^* + p^*) U^* \end{pmatrix}}_{F^*(W^*)} = 0, \tag{10.18}$$

avec l'équation d'état

$$p^* = \rho^* T^*, \tag{10.19}$$

et l'énergie totale

$$E^* = \frac{p^*}{\gamma - 1} + \frac{\rho^*}{2} U^{*2}. \tag{10.20}$$

La définition de la vitesse du son devient :

$$a^* = \sqrt{\gamma \frac{p^*}{\rho^*}} = \sqrt{\gamma T^*}, \tag{10.21}$$

et l'enthalpie totale :

$$H^* = \frac{H}{a_R^2} = \frac{(a^*)^2}{\gamma - 1} + \frac{1}{2} U^{*2}. \tag{10.22}$$

1. Dans le cas général, les invariants sont données sous forme différentielle

$$dr^0 = dp - a^2 d\rho = 0, \quad dr^+ = dp + \rho a dU = 0, \quad dr^- = dp - \rho a dU = 0$$

et doivent être intégrés le long des caractéristiques correspondantes (voir la bibliographie complémentaire).

Pour simplifier la notation, les étoiles seront ignorées par la suite ; toutes les variables s'entendent sans dimension.

10.2.2 Solution exacte

Le problème du tube à choc comporte une solution exacte, obtenue par l'analyse des ondes (choc, détente, ligne de glissement) qui se propagent dans le dispositif (voir la figure 10.1). Dans le plan (x, t), les trois ondes qui font le passage de (R) à (L) séparent deux zones intermédiaires, notées par les indices 1 et 2 (figure 10.2).

> **Remarque 10.1 :** L'analyse des caractéristiques effectuée précédemment montre que la solution exacte $W(x, t)$ ne dépend en fait que d'une seule variable : x/t.

Sans insister sur la signification physique des relations de saut à travers ces ondes (pour plus de détails, voir les ouvrages spécialisés mentionnés à la fin du projet), nous présentons seulement les étapes de construction de la solution exacte.

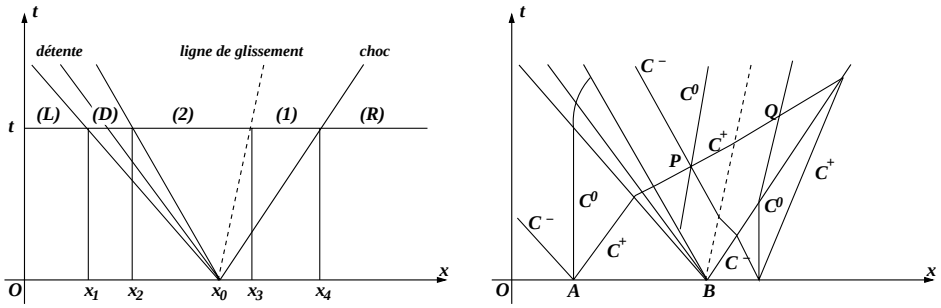


Figure 10.2 Représentation dans le plan (x, t) des ondes qui se propagent dans le tube à choc et des caractéristiques correspondantes.

1. Les données du problème sont les paramètres (grandeurs sans dimension !) des zones (R) et (L) :

$$\text{Zone (R)} \qquad \rho_R = 1, \quad p_R = 1/\gamma, \quad T_R = 1/\gamma, \quad U_R = 0 \tag{10.23}$$

$$\text{Zone (L)} \qquad \rho_L, \quad p_L, \quad T_L, \quad U_L = 0 \tag{10.24}$$

2. Le passage de la zone (R) à la zone (1) se fait par une onde de choc instationnaire, de vitesse constante. Il faut déterminer le nombre de Mach du choc ($M_s = U_s/a_R$), solution de l'équation suivante (appelée relation de compatibilité) :

$$M_s - \frac{1}{M_s} = a_L \frac{\gamma+1}{\gamma-1} \left\{ 1 - \left[\frac{p_R}{p_L} \left(\frac{2\gamma}{\gamma+1} M_s^2 - \frac{\gamma-1}{\gamma+1} \right) \right]^{\frac{\gamma-1}{2\gamma}} \right\}. \tag{10.25}$$

3. Les paramètres de la zone (1) sont constants et donnés par les relations de saut de Rankine-Hugoniot à travers le choc :

$$\frac{p_1}{p_R} = \frac{2\gamma}{\gamma+1}M_s^2 - \frac{\gamma-1}{\gamma+1}, \tag{10.26}$$

$$\frac{\rho_R}{\rho_1} = \frac{2}{\gamma+1}\frac{1}{M_s^2} + \frac{\gamma-1}{\gamma+1}, \tag{10.27}$$

$$U_1 = \frac{2}{\gamma+1}\left(M_s - \frac{1}{M_s}\right). \tag{10.28}$$

   On peut également calculer $T_1 = p_1/\rho_1$ et $a_1 = \sqrt{\gamma T_1}$.

4. Le passage à travers la ligne de glissement est caractérisé par une discontinuité de masse volumique seulement, la vitesse et la pression étant conservées. Par conséquent :

$$U_2 = U_1, \quad p_2 = p_1. \tag{10.29}$$

5. Les paramètres de la zone (2) peuvent être reliés aux paramètres de la zone (L) en utilisant les caractéristiques représentées sur la figure 10.2. En prenant les caractéristiques C^0 et C^+ passant par le point P, on obtient (voir les relations 10.15)

$$\rho_2 = \rho_L\left(\frac{p_2}{p_L}\right)^{1/\gamma}, \quad a_2 = a_L - \frac{\gamma-1}{2}U_2. \tag{10.30}$$

   Remarque 10.2 : Noter que la relation de compatibilité (10.25) à été obtenue en combinant les formules (10.30), (10.29), (10.28) et (10.26).

Il reste à déterminer l'étendue de chaque zone (abscisses x_1, x_2, x_3, x_4 sur la figure 10.2) pour un temps t donné.

– Le faisceau de détente (D) est délimité à gauche par la caractéristique C^- partant du point B, considéré comme appartenant à la zone (L) et à droite par la caractéristique C^- partant du même point B, mais qui appartient cette fois à la zone (2). On peut donc écrire que :

$$x_1 = x_0 - a_L t, \quad x_2 = x_0 + (U_2 - a_2)t. \tag{10.31}$$

  À l'intérieur du faisceau de détente ($x_1 \leqslant x \leqslant x_2$) les paramètres thermodynamiques ne sont plus constants. Un point à l'intérieur de la zone (D) est placé sur une caractéristique C^- partant de B, donc $(x - x_0)/t = U - a$. En utilisant de nouveau une caractéristique C^+ venant de (L), on peut écrire que $a + (\gamma - 1)U/2 = a_L$ et, finalement, la solution exacte dans la zone (D) :

$$U = \frac{2}{\gamma+1}\left(a_L + \frac{x-x_0}{t}\right),\ a = \frac{2}{\gamma+1}\left(a_L - \frac{\gamma-1}{2}\frac{x-x_0}{t}\right),\ p = p_L\left(\frac{a}{a_L}\right)^{\frac{2\gamma}{\gamma-1}}. \tag{10.32}$$

– La ligne de glissement se propage avec la vitesse constante $U_2 = U_1$, donc :

$$x_3 = x_0 + U_2 t. \tag{10.33}$$

– Le choc se déplace aussi à vitesse constante $U_s = M_s a_R$, donc, en variables sans dimension :

$$x_4 = x_0 + M_s t. \tag{10.34}$$

Exercice 10.1

Écrire un fonction qui calcule la solution exacte pour le tube à choc :

```
function uex=tchoc_exact(x,x0,t)
% Arguments d'entrée :
%       x   points de discrétisation (vecteur de dimension n)
%       x0 position initiale de la membrane
%       t    le temps pour lequel la solution exacte est calculée
% Argument de sortie :
%       uex  le vecteur de dimension (3,n) contenant la solution exacte
%           uex(1,1:n)  la masse volumique ρ
%           uex(2,1:n)  la vitesse U
%           uex(3,1:n)  la pression p
```

Tracer la solution exacte pour $x \in [0, 1]$, $n = 81$, $x_0 = 0.5$, $t = 0.2$. Les paramètres physiques (tube à choc de Sod) : $\gamma = 1.4$, $\rho_L = 8$, $p_L = 10/\gamma$ seront définis comme variables globales.

La solution exacte est représentée sur la figure 10.3. Elle est obtenue avec les programmes décrits dans le paragraphe 10.4 (page 227).

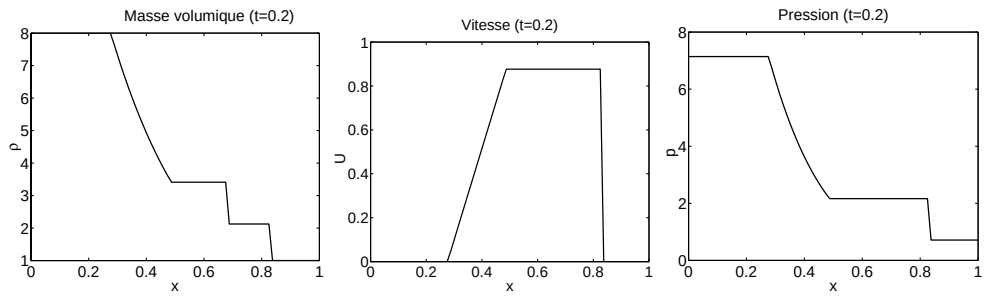


Figure 10.3 Solution exacte du tube à choc de Sod pour $t = 0.2$.

10.3 RÉSOLUTION NUMÉRIQUE

Le système d'Euler (10.18) peut être résolu numériquement en utilisant les méthodes de discrétisation *élémentaires* (voir le Projet 1 pour quelques exemples). En particulier, on peut appliquer une méthode d'Euler (ou de Runge-Kutta) pour l'intégration en temps et des schémas aux différences finies centrées pour la discrétisation spatiale. Nous allons voir cependant que ces méthodes ne sont pas adaptées à la *capture* des discontinuités présentes dans la solution car elle génèrent des oscillations numériques. Ce comportement sera illustré en présentant les schémas centrés de Lax-Wendroff et de MacCormack. Nous donnerons ensuite un exemple de schéma décentré (le schéma de Roe), qui prend en compte le caractère hyperbolique du système d'Euler et permet une meilleure résolution numérique.

10.3.1 Schémas centrés (Lax-Wendroff et MacCormack)

Historiquement, les premiers schémas utilisés pour la résolution des systèmes hyperboliques ont été les schémas centrés de Lax-Wendroff et MacCormack. Ils sont encore utilisés dans certains codes industriels. Considérons d'abord le système (10.18) écrit sous la forme

$$\frac{\partial W}{\partial t} + \frac{\partial}{\partial x} F(W) = 0, \tag{10.35}$$

et la discrétisation habituelle du domaine de définition $(x, t) \in [0, 1] \times [0, T]$:

– en espace

$$x_j = (j-1)\delta x, \quad \delta x = \frac{1}{N_x - 1}, \quad j = 1, 2, \ldots, N_x \tag{10.36}$$

– et en temps

$$t^n = (n-1)\delta t, \quad \delta t = \frac{T}{N_t - 1}, \quad n = 1, 2, \ldots, N_t. \tag{10.37}$$

La solution numérique W_j^{n+1} (pour l'instant t_{n+1} et la position x_j) sera calculée en deux pas, un pas de prédiction et un pas de correction, précisés sur la figure 10.4.

Quelques remarques sur ces deux schémas :

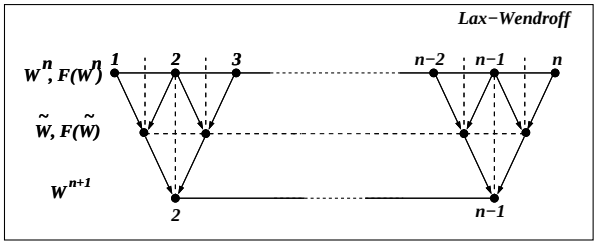
1. Les deux schémas sont explicites. Les deux pas de calcul sont représentés schématiquement sur la figure 10.4 pour chaque méthode. Observons que les schémas permettent de calculer seulement les composantes $j = 2, \ldots, (n-1)$ de la solution. Pour les composantes $j = 1$ et $j = n$ des conditions aux limites doivent être prescrites. Selon l'hypothèse du tube infini, on va imposer $W_1^n = W_L$ et $W_n^n = W_R$ pour chaque instant de temps t_n, ce qui revient pratiquement de garder constantes la première et la dernière composante du vecteur solution.

2. Le schéma de Lax-Wendroff estime dans le pas de prédiction la solution aux interfaces $(j + 1/2)$ et $(j - 1/2)$ en utilisant des différences décentrées en avant pour la dérivée spatiale. Ces valeurs sont utilisées dans le pas de correction pour la discrétisation de la dérivée spatiale avec des différences centrées.

Lax-Wendroff

$$\tilde{W}_{j+1/2} = \frac{W_j^n + W_{j+1}^n}{2} - \frac{\delta t}{2\delta x}\left[F(W_{j+1}^n) - F(W_j^n)\right]$$

$$W_j^{n+1} = W_j^n - \frac{\delta t}{\delta x}\left[F(\tilde{W}_{j+1/2}) - F(\tilde{W}_{j-1/2})\right]$$



MacCormack

$$\tilde{W}_j = W_j^n - \frac{\delta t}{\delta x}\left[F(W_{j+1}^n) - F(W_j^n)\right]$$

$$W_j^{n+1} = \frac{W_j^n + \tilde{W}_j^n}{2} - \frac{\delta t}{2\delta x}\left[F(\tilde{W}_j) - F(\tilde{W}_{j-1})\right]$$

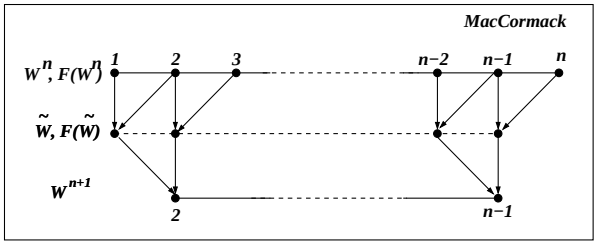


Figure 10.4 Schémas centrés de Lax-Wendroff et MacCormack. Rprésentation schématique du calcul de la solution.

3. Le schéma de MacCormack est basé sur un développement de Taylor ; il construit la solution suivant la relation

$$W_j^{n+1} = W_j^n + \left(\overline{\frac{\partial W}{\partial t}}\right)_j \delta t, \tag{10.38}$$

où

$$\left(\overline{\frac{\partial W}{\partial t}}\right)_j = \frac{1}{2}\left[\left(\frac{\partial W}{\partial t}\right)_j^n + \left(\frac{\partial \tilde{W}}{\partial t}\right)_j\right] = \frac{1}{2}\left[\frac{\tilde{W}_j - W_j^n}{\delta t} - \frac{F(\tilde{W}_j) - F(\tilde{W}_{j-1})}{\delta x}\right] \tag{10.39}$$

est une approximation de la dérivée première en temps.

Observons que des différences décentrées alternées (avant-arrière) sont utilisées dans les deux pas du schéma pour approcher la dérivée spatiale.

4. Les deux schémas utilisent trois points de calcul $(j-1, j, j+1)$ pour atteindre une précision à l'ordre deux en temps et en espace.

5. L'information est cherchée de deux cotés du point de calcul j sans tenir compte de sa propagation par les caractéristiques. Par analogie avec l'équation de convection (voir le Projet 1), on peut dériver une condition de stabilité (ou condition CFL) qui s'écrit de manière générale

$$\max\{|\lambda|\}\cdot\frac{\delta t}{\delta x} \leqslant 1,$$

avec λ valeur propre de la matrice jacobienne $\partial F/\partial W$, interprétée ici comme la vitesse de propagation de l'onde caractéristique associée ($dx/dt = \lambda$). Suivant (10.15), on obtient comme condition de stabilité :

$$\left(|U| + a\right)\frac{\delta t}{\delta x} \leqslant 1. \tag{10.40}$$

En pratique, cette condition sera utilisée pour le calcul du pas de temps sous la forme

$$\delta t = cfl\cdot\frac{\delta x}{|U| + a}, \quad \text{avec} \quad cfl < 1. \tag{10.41}$$

Exercice 10.2

Reprendre l'exercice précédent (même paramètres physiques et numériques) en calculant la solution numérique à $t = 0.2$ par les schémas de Lax-Wendroff et MacCormack. Comparer les résultats avec la solution exacte. Commenter.
Indications :

- le pas de temps sera calculé dans une fonction
  `function dt = calc_dt(w,dx,cfl)` On prendra $cfl = 0.95$.
- utiliser une fonction pour le calcul du vecteur $F(W)$:
  `function f = flux_centre(w)`
- suivre la figure 10.4 pour mettre en œuvre les différents calculs (les boucles `for` peuvent être évitées) ;
- à la fin du calcul, représenter (ρ, U, p), en superposant la solution numérique et la solution exacte.

Les résultats obtenus avec les deux schémas centrés sont présentés sur la figure 10.5. La solution numérique développe des oscillations au voisinage des discontinuités, en particulier au voisinage du choc. L'amplitude des oscillations est plus importante pour le schéma de MacCormack. Le programme qui utilise ce schéma est décrit dans le paragraphe 10.4 (page 227).

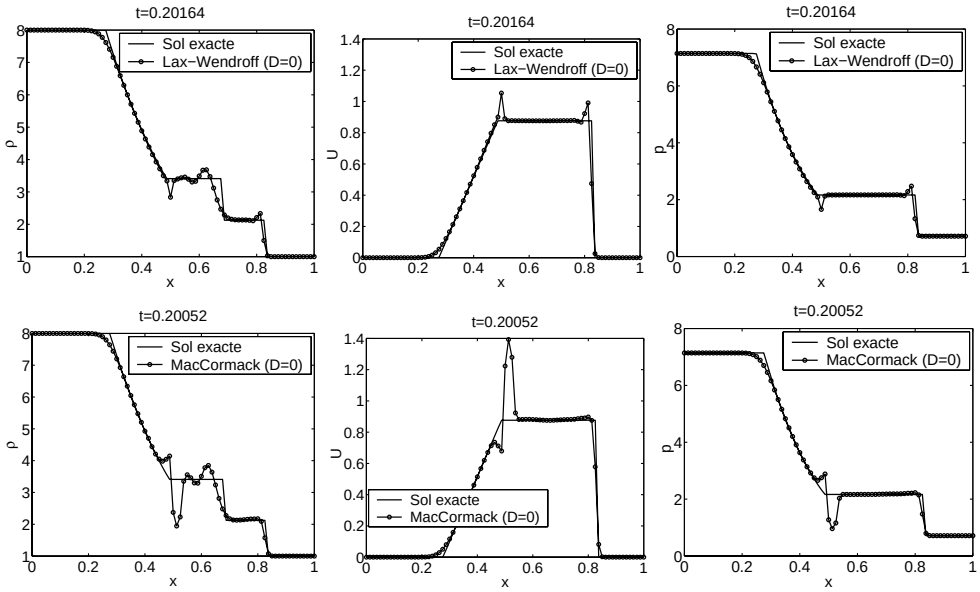


Figure 10.5 Résultats pour le tube à choc de Sod. Calcul avec le schéma de Lax-Wendroff (en haut) et le schéma de MacCormack (en bas).

Dissipation artificielle

La solution numérique obtenue par les schémas centrés présente des oscillations au voisinage des discontinuités. Ces oscillations peuvent être réduites en ajoutant à l'équation initiale (10.35) un terme de dissipation :

$$\frac{\partial W}{\partial t} + \frac{\partial}{\partial x} F(W) - \delta x^2 \frac{\partial}{\partial x}\left(D(x) \frac{\partial W}{\partial x} \right) = 0. \tag{10.42}$$

Le terme de dissipation étant proportionnel au gradient $\partial W/\partial x$, il aura une influence négligeable dans les régions de faible variation de la solution et un effet bien connu de lissage des gradients et d'amortissement des oscillations au niveau des discontinuités (voir l'analyse de l'équation de la chaleur dans le Projet 1).

Les coefficient $D(x)$, également appelé *viscosité artificielle*, doit être positif pour avoir une action stabilisante sur la solution. Plusieurs techniques on été développées pour prescrire la forme de $D(x)$ et corriger la solution W^{n+1} en tenant compte de la dissipation artificielle (voir la bibliographie complémentaire). Nous présentons ici la technique la plus simple qui consiste à prendre $D(x) = D = const$ et à résoudre par les schémas centrés l'équation (10.42) écrite sous la forme :

$$\frac{\partial W}{\partial t} + \frac{\partial}{\partial x} F^*(W) = 0, \quad \text{avec} \quad F^*(W) = F(W) - D\delta x^2 \frac{\partial W}{\partial x}. \tag{10.43}$$

Pour respecter le principe des schémas de Lax-Wendroff et MacCormack (voir figure 10.4), le nouveau vecteur $F^*(W)$ sera discrétisé différemment dans les deux pas :

– pas de prédiction (différences décentrées en arrière)

$$F^*(W_j) = F(W_j) - (D\delta x)(W_j - W_{j-1}) \tag{10.44}$$

– pas de correction (différences décentrées en avant)

$$F^*(\tilde{W}_j) = F(\tilde{W}_j) - (D\delta x)(\tilde{W}_{j+1} - \tilde{W}_j). \tag{10.45}$$

Exercice 10.3

> Ajouter la dissipation artificielle dans les schémas de Lax-Wendroff et MacCormack suivant les équations (10.44)-(10.45). Observer l'influence sur la solution du coefficient de viscosité artificielle D. On prendra $0 \leqslant D \leqslant 10$. Que peut-on dire sur le calcul du pas de temps ?

Les résultats (figure 10.6) montrent que la dissipation artificielle permet de réduire les oscillations, mais cette technique n'apporte pas une solution réelle au problème. Plus la viscosité artificielle est augmentée, plus la solution est détériorée au niveau des autres discontinuités. En particulier, le saut de masse volumique (ρ) caractérisant la ligne de glissement est totalement lissé par la viscosité artificielle. Des techniques plus sophistiquées utilisent une dissipation artificielle sélective, suivant les composantes du vecteur solution (voir la bibliographie complémentaire).

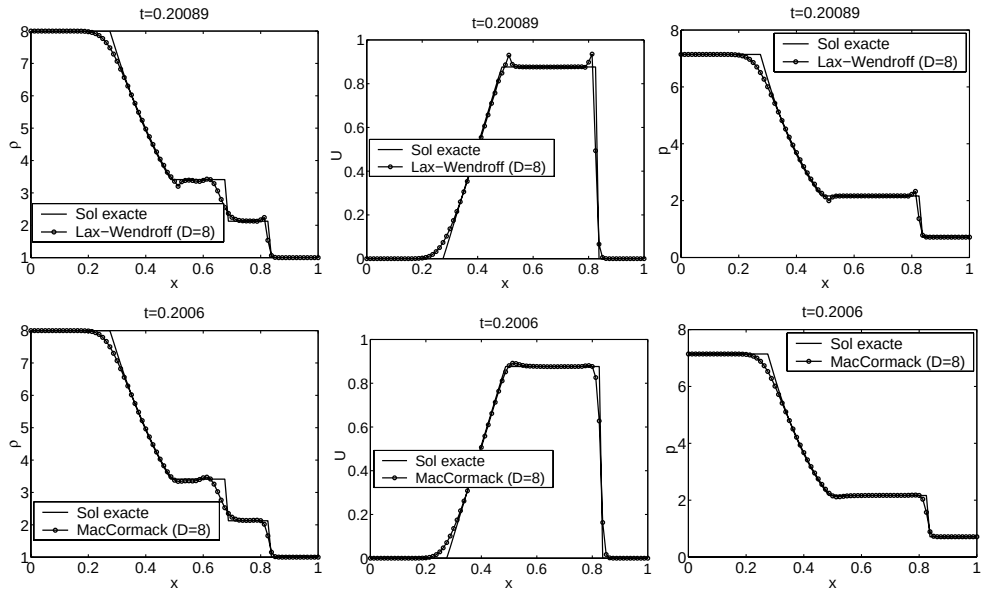


Figure 10.6 Résultats pour le tube à choc de Sod. Calcul avec le schéma de Lax-Wendroff (en haut) et le schéma de MacCormack (en bas) avec un terme de dissipation artificielle.

10.3.2 Schémas décentrés (Roe)

L'origine des oscillations numériques générées par les schémas centrés vient du fait que ces schémas ignorent le caractère hyperbolique de l'écoulement, c'est-à-dire la propagation de l'information par les caractéristiques. Ces informations essentielles sont prises en compte dans la construction des schémas décentrés.

Définition 10.2 *Un schéma est dit décentré (*upwind *en anglais) s'il utilise une discrétisation qui dépend de la direction de propagation de l'onde physique ou du signe de la vitesse de convection des caractéristiques.*

On peut distinguer deux types de schémas décentrés :

1. les schémas de type décomposition du flux (*flux splitting* en anglais) : le flux $F(W)$ est discrétisé différemment, suivant le signe de la vitesse de propagation des ondes caractéristiques, autrement dit, le schéma utilise seulement l'information sur le signe des valeurs propres de la matrice jacobienne (10.8) du système.
2. les schémas de type Godunov : des propriétés dérivées de la solution exacte du système d'Euler sont directement introduites dans la discrétisation.

Nous présentons dans la suite un exemple de schéma de Godunov, le schéma de Roe.

a) Schémas de type Godunov

Dans les schémas de type Godunov la variable W est considérée constante sur chaque *cellule* $]x_{j-1/2}, x_{j+1/2}[$ et l'évolution de la solution en temps est calculée par la résolution exacte du problème de Riemann à l'interface des intervalles élémentaires.

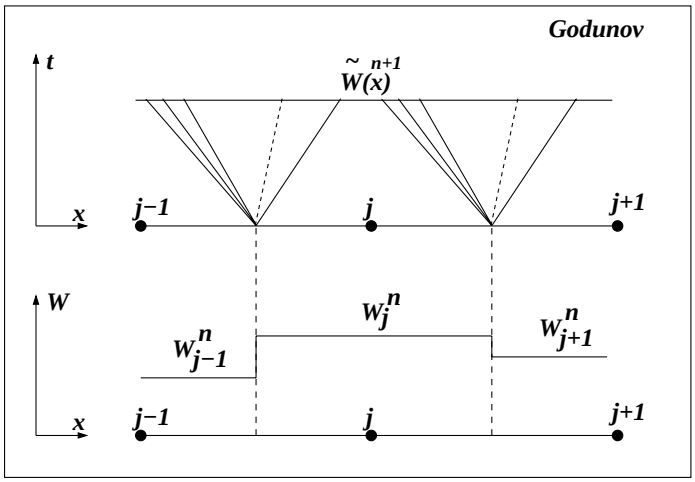


Figure 10.7 Calcul de la solution par le schéma de type Godunov.

Les étapes de construction de la solution numérique sont (voir la figure 10.7) :

Étape 1. Construction de la fonction constante par morceaux

$$W^n(x) = W_j^n, \quad x \in](j-1/2)\delta x, (j+1/2)\delta x[\tag{10.46}$$

Étape 2. Calcul de la solution $\tilde{W}^{n+1}(x)$ en résolvant exactement les problèmes de Riemann aux interfaces $(j-1/2)$ et $(j+1/2)$. Cette étape impose une restriction sur le pas de temps sous la forme

$$\max_j \left(|U| + a\right)_{j+1/2}^n \frac{\delta t}{\delta x} \leqslant 1/2, \tag{10.47}$$

qui interdit aux ondes issues des interfaces voisines de se toucher.

Étape 3. La solution W^{n+1}, constante par morceaux, est obtenue en calculant la moyenne sur chaque cellule :

$$W_j^{n+1} = \frac{1}{\delta x} \int_{(j-1/2)\delta x}^{(j+1/2)\delta x} \tilde{W}^{n+1}(x) dx. \tag{10.48}$$

On peut montrer que le schéma de Godunov peut s'écrire sous la forme suivante, dite *conservative* :

$$\frac{W_j^{n+1} - W_j^n}{\delta t} + \frac{\Phi(W_j^n, W_{j+1}^n) - \Phi(W_j^n, W_{j-1}^n)}{\delta x} = 0, \tag{10.49}$$

avec le vecteur *flux* défini de manière générale par

$$\Phi(W_j^n, W_{j+1}^n) = F(\tilde{W}_{j+1/2}^{n+1}). \tag{10.50}$$

L'avantage de la forme conservative est d'être valable sur tout le domaine de définition, même en présence de discontinuités. La forme exacte du vecteur flux sera précisée pour le schéma de Roe.

b) Schéma de Roe

Parmi les schémas de type Godunov, le schéma de Roe a gagné une grande popularité grâce à sa simplicité et à son ingéniosité. L'idée du schéma est de résoudre à l'interface $j + 1/2$ un problème de Riemann linéaire (10.7), écrit sous la forme :

$$\frac{\partial \tilde{W}}{\partial t} + A_{j+1/2} \frac{\partial \tilde{W}}{\partial x} = 0, \quad \text{avec} \quad \tilde{W}(x, n\delta t) = \begin{cases} W_j^n, & x \leqslant (j+1/2)\delta x \\ W_{j+1}^n, & x > (j+1/2)\delta x \end{cases} \tag{10.51}$$

La difficulté de cette approche est de définir la matrice $A_{j+1/2}$ qui dépend de W_j^n et W_{j+1}^n. De manière générale, elle doit satisfaire aux conditions suivantes :

1. Le caractère hyperbolique du système doit être gardé, donc $A_{j+1/2}$ va s'écrire sous la forme (voir la décomposition 10.12)

$$A_{j+1/2} = P_{j+1/2}\, \Lambda_{j+1/2}\, P_{j+1/2}^{-1} \tag{10.52}$$

Afin de tenir compte du signe de la vitesse de propagation des ondes caractéristiques, il est utile de définir les matrices :

- $\text{sign}(A_{j+1/2}) = P_{j+1/2}\,(\text{sign}(\Lambda))\,P^{-1}_{j+1/2}$, où $\text{sign}(\Lambda)$ est la matrice diagonale, dont la diagonale est formée des signes des valeurs propres λ_l : $\text{sign}(\Lambda) = \text{diag}(\text{sign}\lambda_l)$.
- $|A_{j+1/2}| = P_{j+1/2}|\Lambda|P^{-1}_{j+1/2}$, avec $|\Lambda| = \text{diag}(|\lambda_l|)$.

2. La forme linéaire du problème de Riemann doit être consistante avec le problème initial : pour tout u, on a

$$A_{j+1/2}(u,u) = A(u,u). \tag{10.53}$$

3. Le schéma doit être *conservatif* : pour tous u et v, on a

$$F(u) - F(v) = A_{j+1/2}(u,v)(u-v). \tag{10.54}$$

Une fois la matrice $A_{j+1/2}$ définie, le schéma de Roe peut s'écrire sous la forme conservative (10.49), avec le flux numérique défini par :

$$\Phi(W^n_j, W^n_{j+1}) = \frac{1}{2}\left\{F(W^n_j) + F(W^n_{j+1}) - (\text{sign}(A_{j+1/2}))[F(W^n_{j+1}) - F(W^n_j)]\right\}, \tag{10.55}$$

ou, si on tient compte de (10.54)

$$\Phi(W^n_j, W^n_{j+1}) = \frac{1}{2}\left\{F(W^n_j) + F(W^n_{j+1}) - |A|_{j+1/2}[W^n_{j+1} - W^n_j]\right\}. \tag{10.56}$$

Suivant (10.49), le schéma de Roe s'écrit simplement :

$$W^{n+1}_j = W^n_j - \frac{\delta t}{\delta x}\left[\Phi(W^n_j, W^n_{j+1}) - \Phi(W^n_j, W^n_{j-1})\right]. \tag{10.57}$$

Pour le calcul de la matrice $A_{j+1/2}$, l'idée originale de Roe a été d'exprimer les vecteurs W et $F(W)$ de l'équation (10.18) comme des formes quadratiques des composantes du vecteur $Z = \sqrt{\rho}(1, U, H)$:

$$W = \begin{pmatrix} z_1^2 \\ z_1 z_2 \\ \frac{1}{\gamma} z_1 z_3 + \frac{\gamma-1}{2\gamma} z_2^2 \end{pmatrix}, \quad F(W) = \begin{pmatrix} z_1 z_2 \\ \frac{\gamma-1}{\gamma} z_1 z_3 + \frac{\gamma-1}{2\gamma} z_2^2 \\ z_2 z_3 \end{pmatrix}, \tag{10.58}$$

et d'appliquer la relation générale, valable pour deux formes quadratiques f, g quelconques :

$$(fg)_{j+1} - (fg)_j = \overline{f}(g_{j+1} - g_j) + \overline{g}(f_{j+1} - f_j), \quad \text{avec} \quad \overline{f} = \frac{f_{j+1} + f_j}{2}.$$

On peut donc écrire :

$$\left\{\begin{array}{lcl} W_{j+1} - W_j & = & \overline{B}(Z_{j+1} - Z_j) \\ F(W_{j+1}) - F(W_j) & = & \overline{C}(Z_{j+1} - Z_j) \end{array}\right. \Longrightarrow F(W_{j+1}) - F(W_j) = (\overline{C}\,\overline{B}^{-1})(W_{j+1} - W_j), \tag{10.59}$$

cette dernière relation correspondant exactement à la condition (10.54). La matrice cherchée sera donnée par :

$$A_{j+1/2} = (\overline{C}\,\overline{B})^{-1}. \tag{10.60}$$

Une propriété remarquable de cette matrice (facile à vérifier par calcul direct) est qu'elle peut être calculée à partir de la forme (10.8) en remplaçant les variables (ρ, U, H) par leurs *moyennes de Roe* :

$$\overline{\rho}_{j+1/2} = R_{j+1/2}\rho_j, \quad \overline{U}_{j+1/2} = \frac{R_{j+1/2}U_{j+1} + U_j}{1 + R_{j+1/2}}, \quad \overline{H}_{j+1/2} = \frac{R_{j+1/2}H_{j+1} + H_j}{1 + R_{j+1/2}}, \tag{10.61}$$

$$\overline{a}^2_{j+1/2} = (\gamma - 1)\left(\overline{H}_{j+1/2} - \frac{\overline{U}^2_{j+1/2}}{2}\right), \quad \text{avec} \quad R_{j+1/2} = \sqrt{\frac{\rho_{j+1}}{\rho_j}}.$$

Les formules des vecteurs et valeurs propres (10.10), (10.11) s'appliquent également pour la matrice $A_{j+1/2}$ en prenant les moyennes de Roe (10.61). Ces relations remarquables simplifient considérablement le calcul des flux numériques dans (10.56), ce qui a fait le succès du schéma.

Remarque 10.3 : Le schéma de Roe est d'ordre un en temps et en espace.

Exercice 10.4

> Résoudre le problème du tube à choc en appliquant le schéma de Roe (10.57). Comparer avec les résultats obtenus avec les schémas centrés.

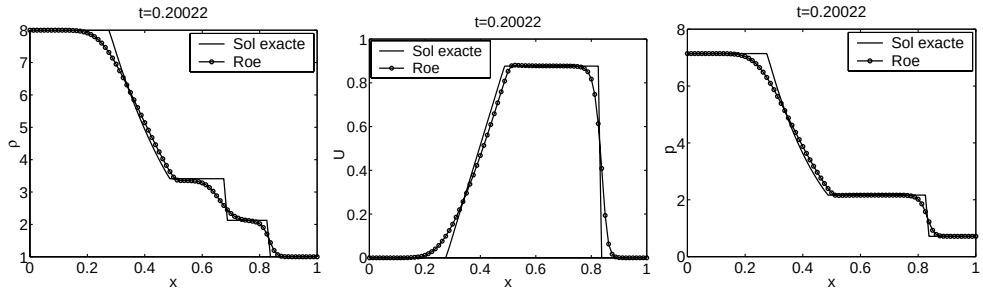


Figure 10.8 Résultats pour le tube à choc de Sod. Calcul avec le schéma de Roe.

Les résultats numériques obtenus (voir la figure 10.8) montrent que le schéma de Roe permet de bien capturer les chocs, mais il s'avère très dissipatif au niveau des autres types de discontinuités (onde de détente et ligne de glissement). Des méthodes plus performantes peuvent être développées dans le cadre des schémas de type Godunov en augmentant la précision en espace. Par exemple, on pourrait utiliser des fonctions linéaires par morceaux pour approcher la solution dans les étapes 1 et 3 du

schéma de Godunov et obtenir des schémas à l'ordre deux. Les schémas utilisés actuellement pour résoudre le système d'Euler incluent plus d'informations physiques dans leur structure (conservation d'entropie, conservativité, etc.) afin de mieux résoudre les discontinuités. La description de ces schémas dépasse le cadre de cette présentation *introductive* ; le lecteur intéressé est dirigé vers les ouvrages spécialisés mentionnés dans la bibliographie [Fletcher, 1991, Hirsh, 1988, LeVeque, 1992, Saad, 1998].

10.4 SOLUTIONS ET PROGRAMMES

Commençons par présenter le script MATLAB *tchoc_ex.m* contenant la fonction qui calcule la solution exacte pour un temps t donné. La relation de compatibilité (10.25) est implémentée sous forme de fonction dans le script *mach_choc.m*. Cette dernière fonction sera utilisée comme argument d'entrée dans la commande MATLAB `fzero` pour calculer la racine M_s de la relation de compatibilité. La solution exacte qui contient les valeurs de (ρ, U, p) est ensuite calculée suivant les relations mathématiques exposées. Remarquons juste l'utilisation de la commande MATLAB `find` pour séparer les différentes zones de l'écoulement.

Le programme principal pour le projet entier est dans le fichier *tchoc.m*. Après la définition des variables globales (qui sont essentiellement les paramètres des zones (R) et (L)) et la discrétisation du domaine spatial, la solution est initialisée avec les données du tube à choc de Sod. Les calculs utilisent trois vecteurs principaux :

- `usol` pour la solution finale (ρ, U, p) ;
- `w` pour la variable $W = (\rho, \rho U, E)$;
- `F` pour la variable $F(W)$, calculée à partir de W.

Le programme permet de choisir parmi les trois schémas implémentés ; quand un schéma centré est sélectionné, le coefficient de viscosité artificielle doit être introduit à la console. La solution numérique est tracée sur les mêmes graphiques que la solution exacte en utilisant la fonction `plot_graph` qui se trouve dans le fichier *plot_graph.m* .

Les principales fonctions utilisées par le programme principal sont les suivantes (elles ont été écrites dans un souci de lisibilité du programme par rapport aux formules mathématiques) :

- *trans_usol_w.FF* : calcul de $W = (\rho, \rho U, E)$ à partir de *usol* $= (\rho, U, p)$;
- *trans_w_usol.FF* : calcul de *usol* $= (\rho, U, p)$ à partir de $W = (\rho, \rho U, E)$;
- *trans_w_f.FF* : calcul de $F = (\rho U, \rho U^2 + p, (E + p)U)$ à partir de $W = (\rho, \rho U, E)$;
- *calc_dt.FF* : calcul de $\delta t = cfl \cdot \delta x/(|U| + a)$ à partir de $W = (\rho, \rho U, E)$.

Revenons maintenant à la programmation des schémas numériques. Les capacités de calcul vectoriel de MATLAB ont été exploitées pour la plupart des calculs.

Ce type d'écriture compacte permet de faire plus facilement l'analogie entre les lignes de programme et les relations analytiques. Par exemple, pour le schéma de MacCormack, on utilise comme guide pour la programmation la figure 10.4. Le vecteur $F(W)$ sera calculé pour tous les points $j = 1, \ldots, n$, tandis que la solution intermédiaire $\tilde{W}$ est évaluée seulement aux points $j = 1, \ldots, n-1$. On peut éviter les boucles `for` pour ce calcul en écrivant

```
wtilde=0.5*(w(:,1:n-1)+w(:,2:n))-0.5*dt/dx*(F(:,2:n)-F(:,1:n-1));
```

ce qui est en fait l'équivalent du pas de prédiction (voir la figure 10.4), effectué pour tous les $j = 1, \ldots, n-1$. La même technique est utilisée pour le pas de correction, le vecteur solution W^{n+1} ayant seulement les composantes $j = 2, \ldots n-1$ calculées par le schéma numérique.

La dissipation artificielle est introduite en utilisant les relations (10.44) et (10.45) sous forme vectorielle ; la commande MATLAB `diff` est utilisée pour calculer les différences $W_{j+1} - W_j$. Suivant les hypothèses sur les conditions aux limites, le vecteur contenant le terme de dissipation est complété par des zéros pour $j = 1$ (pas de prédiction) et, respectivement, $j = n-1$ (pas de correction).

Une attention particulière a été accordée au schéma de Roe qui nécessite un calcul plus laborieux du flux numérique Φ. Ce calcul est effectué dans la fonction `flux_roe.m`. Afin de réduire l'encombrement mémoire, la partie du flux évaluée aux interfaces $(j + 1/2)$ a été calculée en utilisant une boucle `for` et plusieurs variables locales facilement identifiables par rapport aux relations analytiques. Remarquons également que la formule analytique de $P^{-1}_{j+1/2}$ a été préférée à l'utilisation de la commande MATLAB `inv` (calcul de l'inverse d'une matrice).

BIBLIOGRAPHIE

[Bonnet et Luneau, 1989] A. Bonnet, J. Luneau, *Aérodynamique : Théories de la Dynamique des Fluides*, Editions Cépaduès, 1989.

[Fletcher, 1991] C. A. J. Fletcher : *Computational Techniques for Fluid Dynamics*, Springer-Verlag, 1991.

[Godlewski et Raviart, 1996] E. Godlewski et P.-A. Raviart : *Numerical approximation of hyperbolic systems of conservation laws*, Springer Verlag, 1996.

[Hirsh, 1988] C. Hirsch : *Numerical computation of internal and external flows*, John Wiley & Sons, 1988.

[LeVeque, 1992] R. LeVeque : *Numerical Methods for Conservation Laws*, Birkhäuser, 1992.

[Saad, 1998] M. Saad, *Compressible Fluid Flow*, Pearson Education, 1998.

Projet **11**

Thermique : optimisation de la température d'un four

Fiche du projet

Difficulté :	3
Notions développées :	Éléments finis 2D, Laplacien, problème direct, problème inverse
Domaines d'application :	Thermique, optimisation

Ce chapitre est consacré à l'étude d'un exemple simple mais réaliste d'un problème d'optimisation. Il s'agit de déterminer la température dans un four destiné à la cuisson d'une pièce de résine thermo–formée. Les éléments chauffants sont des résistances électriques. À partir de la valeur connue de chacune des résistances, on cherche à calculer le champ de température à l'intérieur du four, et en particulier la température de l'objet à cuire. Sachant que la température idéale de cuisson conditionne la solidité du produit fini et que des essais expérimentaux sont longs et onéreux, cette simulation numérique permet de vérifier à moindre coût la validité des réglages du four.

Cette première approche est appelée problème direct : on calcule la température de l'objet en fonction de la valeur des résistances. Mais la démarche réelle consiste à déterminer la valeur des résistances, considérées comme inconnues, en fonction de la température idéale de cuisson de l'objet, qui est alors la donnée du problème. Cette approche, appelée problème inverse, est un problème d'optimisation qui est traité en fin de chapitre.

> **Remarque 11.1 :** Pour le calcul du champ de température on utilise la méthode des éléments finis. Seules les grandes lignes de la méthode, indispensables à la compréhension de l'exposé, sont reprises dans ce chapitre.

11.1 FORMULATION DU PROBLÈME

Le four est représenté par un domaine Ω de frontière $\partial\Omega$ (figure 11.1). Soit $\partial\Omega_D$ une partie de $\partial\Omega$ de mesure non nulle, on note $\partial\Omega_N$ et $\partial\Omega_F$ deux parties de la frontière telles que

$$\partial\Omega = \partial\Omega_D \cup \partial\Omega_N \cup \partial\Omega_F \quad \text{et} \quad \partial\Omega_D \cap \partial\Omega_N = \partial\Omega_D \cap \partial\Omega_F = \partial\Omega_F \cap \partial\Omega_N = \emptyset.$$

On écrit alors l'équation de diffusion de la chaleur sous la forme

$$\begin{cases} \text{Trouver } T \in V, \text{ tel que} \\ -div\,(\mathbb{K}\,\overrightarrow{\text{grad}}\,T) = F \text{ dans } \Omega. \end{cases} \tag{11.1}$$

La température du four est imposée sur une partie du bord : $T = T_D$ sur $\partial\Omega_D$. Ce type de condition est appelée *condition aux limites de Dirichlet.* Une condition sur $\partial\Omega_N$ régule les échanges thermiques avec l'extérieur. Ce type de condition est appelée *condition aux limites de Neumann.* Enfin une condition de Fourier sur $\partial\Omega_F$ indique que les échanges thermiques entre le four et l'extérieur sont proportionnels à la différence de température $T - T_F$.

Ces hypothèses physiques se traduisent par des conditions mathématiques associées au problème (11.1). On les appelle *conditions aux limites* :

$$\begin{cases} T = T_D \text{ sur } \partial\Omega_D \\ \sum_{i,j} \mathbb{K}_{i,j} \dfrac{\partial T}{\partial x_j} \nu_i = f \text{ sur } \partial\Omega_N \\ \sum_{i,j} \mathbb{K}_{i,j} \dfrac{\partial T}{\partial x_j} \nu_i = g(T - T_F) \text{ sur } \partial\Omega_F. \end{cases} \tag{11.2}$$

Dans cette formulation on a retrouvé

1. La température T dans le domaine Ω.
2. L'ensemble V des températures admissibles.

3. Le tenseur de conductivité thermique $\mathbb{K} \in \mathbb{R}^{2\times 2}$. Dans un milieu homogéne et isotrope $\mathbb{K} = cI_2$ avec c coefficient de conductivité.
4. Les sources de chaleur volumiques F et les sources de chaleur surfaciques f .
5. La température T_D fixée à la frontière $\partial\Omega_D$.
6. La température T_F, température extérieure à la frontière $\partial\Omega_F$.
7. Le coefficient d'échange thermique g à la frontière $\partial\Omega_F$.
8. Le vecteur $\vec{\nu} = (\nu_1, \nu_2)^T$, normale extérieure à $\partial\Omega$.

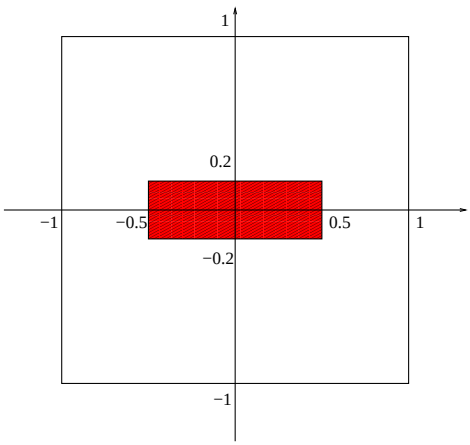


Figure 11.1 L'objet dans le four

Pour simplifier le problème, on traitera le cas d'une isolation thermique parfaite, qui se traduit par la condition $f = 0$ sur $\partial\Omega_N$ et $g = 0$ sur $\partial\Omega_F$. Le problème physique est maintenant entièrement déterminé. Pour obtenir le problème mathématique correspondant on utilise la formule de Green

$$\forall T, T' \in V$$
$$-\int_\Omega div\left[\mathbb{K}\,\overrightarrow{\text{grad}}\,T\right]T'dx = \sum_{i,j}\int_\Omega \mathbb{K}_{i,j}\frac{\partial T}{\partial x_j}\frac{\partial T'}{\partial x_i}\,dx - \sum_{i,j}\int_{\partial\Omega}\mathbb{K}_{i,j}\frac{\partial T}{\partial x_j}T'\,\nu_i\,ds.$$

On introduit alors le sous-espace $V^0 \subset V$ par $V^0 = \{T' \in V, T'\,|_{\partial\Omega_D} = 0\}$ et on écrit la formulation variationnelle du problème de thermique

$$\begin{cases} \text{Trouver } T \in V^0 + T_D \text{ tel que} \\ \forall T' \in V^0, \quad \displaystyle\sum_{K\in\mathcal{T}_h}\int_K \overrightarrow{\text{grad}}^t\,T'\,\mathbb{K}\,\overrightarrow{\text{grad}}\,T\,dx = \sum_{K\in\mathcal{T}_h}\int_K T'\,F\,dx. \end{cases} \tag{11.3}$$

On peut démontrer que le problème (11.3) est équivalent au problème physique (11.1)+(11.2) et qu'il admet la même solution unique T.

11.2 DISCRÉTISATION PAR ÉLÉMENTS FINIS

La forme du domaine Ω, l'expression des données physiques du problème (souvent mesurées expérimentalement) ne permettent pas dans le cas général de trouver une solution du problème (11.1)+(11.2) sous la forme explicite $T(x, y)$. Cette solution ne peut alors être obtenue que par valeurs approchées, à l'aide par exemple de la méthode des éléments finis. Cette méthode consiste à découper le domaine Ω en éléments triangulaires et à représenter la solution T dans chaque triangle par un polynôme dont on doit calculer les coefficients. On appelle triangulation $\mathcal{T}_h$ du domaine Ω un ensemble de triangles répondant aux conditions suivantes

$$\overline{\Omega} = \bigcup_{K \in \mathcal{T}_h} K.$$

$$\forall K, K' \in \mathcal{T}_h, \qquad K \cap K' = \begin{cases} \emptyset, \text{ ou} \\ \text{un sommet commun à } K \text{ et } K', \text{ ou} \\ \text{un côté commun à } K \text{ et } K'. \end{cases}$$

À l'aide de la triangulation $\mathcal{T}_h$, on construit V_h sous–espace vectoriel de dimension finie de V. Pour cette construction, le choix le plus simple correspond aux éléments finis de Lagrange de degré 1. Dans un triangle quelconque K de $\mathcal{T}_h$, de sommets A_1, A_2 et A_3, un élément T'_h de V_h s'écrit

$$T'_h(x, y) = T'_h(A_1)\lambda_1 + T'_h(A_2)\,\lambda_2 + T'_h(A_3)\lambda_3 \tag{11.4}$$

où λ_i est la $i^{ième}$ coordonnée barycentrique du point M de coordonnées (x, y) dans le triangle K. On rappelle que les coordonnées barycentriques du point M dans le triangle K, de sommets A_1, A_2 et A_3 sont les trois réels λ_1, λ_2 et λ_3 (uniques si A_1, A_2 et A_3 ne sont pas alignés) vérifiant

$$\lambda_1 + \lambda_2 + \lambda_3 = 1$$

et

$$\overrightarrow{OM} = \lambda_1\overrightarrow{OA}_1 + \lambda_2\overrightarrow{OA}_2 + \lambda_3\overrightarrow{OA}_3.$$

Les coordonnées cartésiennes (x, y) d'un point M sont liées à ses coordonnées barycentriques $(\lambda_1, \lambda_2, \lambda_3)$ dans le triangle de sommets A_1, A_2 et A_3 par les relations

$$\begin{cases} x(M) &= x(A_3) + [x(A_1) - x(A_3)]\lambda_1 + [x(A_2) - x(A_3)]\lambda_2 \\ y(M) &= y(A_3) + [y(A_1) - y(A_3)]\lambda_1 + [y(A_2) - y(A_3)]\lambda_2. \end{cases} \tag{11.5}$$

Avec ces notations, la définition (11.4) s'écrit encore

$$T'_h(x, y) = T'_h(A_3) + [T'_h(A_1) - T'_h(A_3)]\lambda_1 + [T'_h(A_2) - T'_h(A_3)]\lambda_2.$$

On définit alors le sous–espace $V_h^0 \subset V_h$

$$V_h^0 = \{T' \in V_h, T' \mid_{\partial\Omega_D} = 0\}.$$

Soit T_D un élément de V_h défini par ses valeurs $T_D(A_l)$ aux sommets de la frontière $\partial\Omega_D$ - elles sont données par les conditions (11.2) - et qui est nul aux autres sommets de $\mathcal{T}_h$. La formulation variationnelle approchée du problème (11.3) s'écrit

$$\left\{ \begin{array}{l} \text{Trouver } T_h \in T_D + V_h^0 \text{ tel que} \\ \forall T'_h \in V_h^0, \quad \displaystyle\sum_{K\in\mathcal{T}_h} \int_K \overrightarrow{\text{grad}}^t\, T'_h\, \mathbb{K}\, \overrightarrow{\text{grad}}\; T_h\, dx = \sum_{K\in\mathcal{T}_h} \int_K T'_h\, F\, dx. \end{array} \right. \tag{11.6}$$

11.3 MISE EN ŒUVRE

La formulation variationnelle (11.6) fait intervenir des intégrales définies sur les triangles de $\mathcal{T}_h$. Afin de détailler les calculs à réaliser, examinons ces termes pour un élément quelconque K. Il faut évaluer

$$\int_K \overrightarrow{\text{grad}}^t\, T'_h\, \mathbb{K}\, \overrightarrow{\text{grad}}\; T_h\, dx \quad \text{et} \quad \int_K F T'_h\, dx.$$

11.3.1 Calcul de la matrice

Pour tout élément T'_h de V_h, il faut calculer $\overrightarrow{\text{grad}}\; T'_h$. Toute fonction $T'_h \in V_h$ vérifie

$$\frac{\partial T'_h}{\partial \lambda_i} = \frac{\partial T'_h}{\partial x} \times \frac{\partial x}{\partial \lambda_i} + \frac{\partial T'_h}{\partial y} \times \frac{\partial y}{\partial \lambda_i} \quad \text{pour } i = 1, 2. \tag{11.7}$$

Mais d'après les relations (11.4) et (11.5)

$$\left| \begin{array}{ll} \frac{\partial T'_h}{\partial \lambda_1} & = T'_h(A_1) - T'_h(A_3) \\ \frac{\partial T'_h}{\partial \lambda_2} & = T'_h(A_2) - T'_h(A_3) \end{array} \right., \quad \left| \begin{array}{ll} \frac{\partial x}{\partial \lambda_1} & = x(A_1) - x(A_3) \\ \frac{\partial x}{\partial \lambda_2} & = x(A_2) - x(A_3) \end{array} \right. \text{ et } \left| \begin{array}{ll} \frac{\partial y}{\partial \lambda_1} & = y(A_1) - y(A_3) \\ \frac{\partial y}{\partial \lambda_2} & = y(A_2) - y(A_3) \end{array} \right. \tag{11.8}$$

On en déduit une nouvelle formulation de (11.7)

$$\begin{bmatrix} T'_h(A_1) - T'_h(A_3) \\ T'_h(A_2) - T'_h(A_3) \end{bmatrix} = \begin{bmatrix} x(A_1) - x(A_3) & y(A_1) - y(A_3) \\ x(A_2) - x(A_3) & y(A_2) - y(A_3) \end{bmatrix} \begin{bmatrix} \dfrac{\partial T'_h}{\partial x} \\ \dfrac{\partial T'_h}{\partial y} \end{bmatrix}. \tag{11.9}$$

Le déterminant de la matrice de (11.9) est

$$\Delta = [x(A_1) - x(A_3)] \times [y(A_2) - y(A_3)] - [x(A_2) - x(A_3)] \times [y(A_1) - y(A_3)].$$

Sa valeur absolue est égale à deux fois la surface du triangle K. La matrice de (11.9) est donc inversible. On pose alors

$$[dl\ T_h']_K = \begin{bmatrix} T_h'(A_1) \\ T_h'(A_2) \\ T_h'(A_3) \end{bmatrix} \text{ et } B = \frac{1}{\Delta} \begin{bmatrix} y(A_2)-y(A_3) & y(A_3)-y(A_1) & y(A_1)-y(A_2) \\ x(A_3)-x(A_2) & x(A_1)-x(A_3) & x(A_2)-x(A_1) \end{bmatrix}.$$

On obtient la relation

$$\int_K (\overrightarrow{\text{grad}}\ T_h')^T\ \mathbb{K}\ \overrightarrow{\text{grad}}\ \ T_h\ dx = [dl\ T_h']_K^T [A_K][dl\ T_h]_K$$

dans laquelle la matrice "élémentaire" $[A_K]$ vérifie

$$[A_K] = \frac{1}{2} c(K) \Delta B^T B.$$

Le coefficient de conductivité dans le triangle K, noté $c(K)$, est distinct pour l'air et la résine.

11.3.2 Calcul du second membre

Supposons pour simplifier que la source de chaleur F est constante par élément. On prend donc $F = 1$ si le triangle K contient une résistance, 0 sinon. En utilisant les mêmes notations on écrit

$$\int_K T_h' F\ dx = [dl\ T_h']_K^T [b_K]$$

avec

$$[b_K] = \frac{1}{6} F(K) \Delta \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}.$$

11.3.3 Le système linéaire

En regroupant tous ces résultats le problème (11.6) s'écrit sous la forme

$$\begin{cases} \text{Trouver } T_h \in T_D + V_h^0 \text{ tel que} \\ \forall T_h' \in V_h^0, \quad \displaystyle\sum_{K \in \mathcal{T}_h} [dl\ T_h']_K^T [A_K][dl\ T_h]_K = \sum_{K \in \mathcal{T}_h} [dl\ T_h']_K^T [b_K]. \end{cases}$$

Dans cette expression $[A_K]$ est la matrice élémentaire et $[b_K]$ est le second membre élémentaire. Le vecteur $[dl\ T_h']_K$ représente les valeurs de la température aux sommets d'un triangle quelconque K de la triangulation $T_h' \in V_h$. Lorsque K parcourt $\mathcal{T}_h$ tous les éléments T_h' de V_h sont pris en compte et on écrit

$$\begin{cases} \text{Trouver } T_h \in T_D + V_h^0 \text{ tel que} \\ \forall T_h' \in V_h^0, \qquad [dl\ T_h']^T\ [A]\ [dl\ T_h] = [dl\ T_h']^T\ [b]. \end{cases} \tag{11.10}$$

Si on note *ns* le nombre de sommets de la triangulation $\mathcal{T}_h$, $[A]$ est une matrice de $\mathbb{R}^{ns\times ns}$ et $[b]$ un vecteur de R^{ns} (*ns* est le nombre de sommets de $\mathcal{T}_h$). De même

$$[dl\ T'_h]^T = [T'_h(A_1), T'_h(A_2), \ldots, T'_h(A_{ns})]^T \text{ et } [dl\ T_h]^T = [T_h(A_1), T_h(A_2), \ldots, T_h(A_{ns})]^T$$

sont des vecteurs de $\mathbb{R}^{ns}$. En conclusion (11.6) est un système linéaire d'ordre *ns*

$$[A]\ [dl\ T_h] = [b]. \tag{11.11}$$

11.4 PRISE EN COMPTE DES CONDITIONS AUX LIMITES

Il faut maintenant prendre en compte la condition aux limites contenue dans la définition de l'espace V_h^0, à savoir la condition $T'_h = 0$ sur Ω_D. Supposons pour simplifier l'écriture que les *nsd* sommets de $\mathcal{T}_h$ situés sur Ω_D sont numérotés en dernier dans la numérotation globale des sommets de $\mathcal{T}_h$. Un vecteur T'_h appartient à V_h^0 si et seulement si ses *nsd* dernières composantes sont nulles. À la lecture de (11.10) on constate que le système linéaire résultant (11.11) ne comporte plus que $(ns - nsd)$ lignes ! Mais comme la solution T_h appartient à $T_D + V_h^0$, ses *nsd* dernières composantes sont connues. Le système linéaire (11.11) n'a que $(ns - nsd)$ inconnues : le compte est bon ! Pour prendre en compte les conditions aux limites $T_h = T_D$ sur Ω_D, le système linéaire à résoudre doit être modifié. Il est d'abord écrit sous la forme

$$\begin{bmatrix} A_1 & A_2 \\ A_2^T & A_3 \end{bmatrix} \times \begin{bmatrix} T_h \\ T_{hD} \end{bmatrix} = \begin{bmatrix} b \\ c \end{bmatrix}.$$

La matrice A_1 est d'ordre $(ns - nsd)$, A_2 est une matrice à $(ns - nsd)$ lignes et *nsd* colonnes et A_3 est une matrice d'ordre *nsd*. La prise en compte de la condition $T'_h\ |_{\partial\Omega_D} = 0$ supprime les *nsd* dernières lignes du système linéaire. Ces lignes sont remplacées par les *nsd* relations $T_h\ |_{\partial\Omega_D} = T_{hD} = T_D$.

Le système linéaire s'écrit maintenant

$$\begin{bmatrix} A_1 & A_2 \\ 0 & I \end{bmatrix} \times \begin{bmatrix} T_h \\ T_{hD} \end{bmatrix} = \begin{bmatrix} b \\ T_D \end{bmatrix}.$$

La matrice A_1 est d'ordre $(ns - nsd)$, la matrice A_2 a $(ns - nsd)$ lignes et *nsd* colonnes et I est la matrice identité d'ordre *nsd*. Si on veut exploiter la symétrie du problème initial pour réduire la taille de la matrice du système linéaire à résoudre, une dernière modification est nécessaire. On élimine le bloc A_2 dans la matrice précédente pour obtenir

$$\begin{bmatrix} A_1 & 0 \\ 0 & I \end{bmatrix} \times \begin{bmatrix} T_h \\ T_{hD} \end{bmatrix} = \begin{bmatrix} b - A_2 T_D \\ T_D \end{bmatrix}.$$

C'est ce système linéaire qui est résolu dans l'ordinateur.

Résolution numérique du problème

La première étape dans la résolution d'un problème par la méthode des éléments finis est de créer le maillage sur lequel on définit l'approximation de la solution. De nombreux logiciels permettent de créer facilement des maillages en dimension 2. Par exemple le maillage de la figure 11.2 a été créé par le code *emc2* de l'INRIA. Il comporte 304 triangles et 173 sommets. Le maillage de la figure 11.3 a été créé à l'aide de la "boîte à outils" PDE-tool de MATLAB . Il comporte 1392 triangles et 732 sommets.

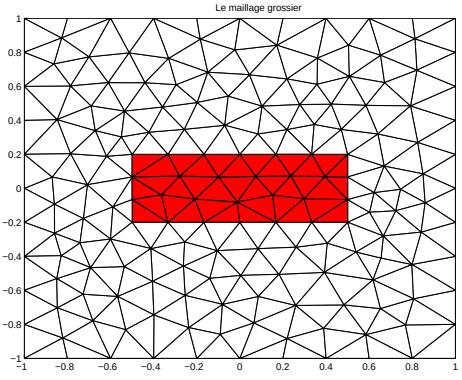


Figure 11.2 Le maillage grossier

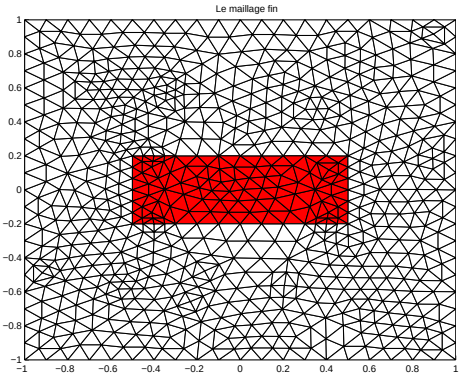


Figure 11.3 Le maillage fin

La procédure qui relit un fichier de maillage dépend de la façon dont il a été créé, mais les informations nécessaires à la mise en œuvre de la méthode des éléments finis sont toujours de la forme

- Nbpt, Nbtri : le nombre de sommets, le nombre de triangles (deux entiers)
- la liste des sommets : pour Ns=1,Nbpt
- Ns Coorneu[Ns,1] Coorneu[Ns,2] Refneu[Ns] : numero du sommet, ses coordonnées et sa référence (numéro de frontière) (un entier, deux réels, un entier)
- la liste des triangles : pour Nt=1,Nbtri
- Nt Numtri[Nt,1 :3] Reftri[N] : numero du triangle, numeros de ses sommets et sa référence (numéro du milieu) (cinq entiers)

Exercice 11.1

1. Créer un maillage (ou relire un des fichiers) et vérifier le contenu des tableaux `Coorneu` et `Numtri`.
2. Calculer les tableaux élémentaires dans chaque triangle.
3. Écrire une procédure qui assemble le système linéaire à partir des tableaux élémentaires.

4. Modifier le système linéaire pour prendre en compte les conditions aux limites.
5. Résoudre le système linéaire résultant.
6. Visualiser les résultats sous forme de courbes isothermes.

Conseil : Pour l'assemblage de la matrice A, utiliser l'algorithme formel suivant :

```
for K=1:Nbtri
   a) lecture des donnees dans le triangle K
      XY   = coordonnees des sommets du triangle  K
      NUM  = numeros des sommets du triangle  K
   b) Calcul des tableaux elementaires AK(3,3) et bK(3)
   c) Assemblage de la matrice  Ktotal(Nbpt,Nbpt)
      for i=1:3
      for j=1:3
         Ktotal(Num(i),Num(j)) = Ktotal(Num(i),Num(j))
                                              + AK(i,j)
      end
      end
   d) Assemblage du second membre  Smb(Nbpt)
      for i=1:3
         Smb(Num(i)) = Smb(Num(i)) + bK(i)
      end
end
```

La solution de cet exercice est présentée à la page 242.

Un champ de température est présenté sur la figure 11.4. On y voit les variations de la température dans le domaine Ω pour une température imposée $T_D = 50$ degrés C sur le bord supérieur du four ($y = 1$) et $T_D = 100$ degrés sur le bord inférieur ($y = -1$), sans résistance chauffante. Un autre champ de température est présenté sur la figure 11.5 : il s'agit des variations de la température avec les conditions aux limites précédentes plus quatre résistances chauffantes (chacune de valeur 25 000). On constate que la température obtenue dans l'objet est très éloignée de la température idéale de cuisson, qui vaut dans notre exemple 250 degrés. Pour corriger ce défaut il faut bien sûr augmenter la valeur des résistances ; mais de combien ? Pour réaliser ce calcul, nous sommes partis de valeurs connues des résistances pour calculer la température de l'objet. C'est ce que l'on appelle un problème direct : les valeurs des résistances sont les données du problème, la température de l'objet en est l'inconnue. Dans la réalité les ingénieurs ne procèdent pas de cette manière. Puisque la qualité du produit fini dépend de la température idéale de cuisson, il faut déterminer les valeurs des résistances qui permettent de chauffer l'objet à cette température. Il s'agit alors d'un problème inverse : la température de l'objet est une donnée du problème, les valeurs des résistances en sont les inconnues.

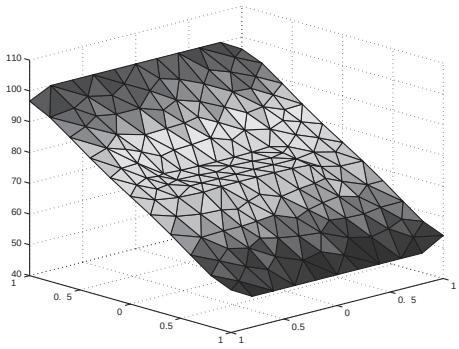


Figure 11.4 La température sans résistance

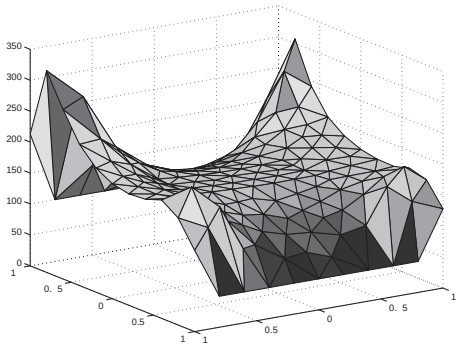


Figure 11.5 La température avec 4 résistances

11.5 FORMULATION DU PROBLÈME INVERSE

Le problème (11.1-11.2) a une propriété importante, il est dit *linéaire* au sens suivant : si T' est la solution du problème correspondant aux données $\{F', T'_D, f'\}$ et T'' la solution du problème correspondant aux données $\{F'', T''_D, f''\}$, alors $T' + T''$ est la solution du problème correspondant aux données $\{F' + F'', T'_D + T''_D, f' + f''\}$. Pour déterminer les valeurs des résistances qui permettent de chauffer l'objet à la température idéale de cuisson, on va utiliser cette propriété. La température T du four pour une température aux bords T_D et un flux de chaleur f imposés à la frontière et avec nr résistances s'écrit

$$T = T_0 + \sum_{k=1}^{nr} \alpha_k T_k$$

où α_k est la valeur de la résistance k et T_k le champ de température associé quand la résistance k est seule à chauffer le four. Pour obtenir la température idéale T_{opt} dans l'objet à cuire S il faut donc déterminer les valeurs α_k des résistances. On les obtient par minimisation de la fonctionnelle

$$J(\alpha) = \int_S \left[T_{opt}(x) - T_0(x) - \sum_{k=1}^{nr} \alpha_k \, T_k(x)\right]^2 dx.$$

La fonctionnelle J est strictement convexe en α. Elle atteint son unique minimum pour la valeur de α qui annule son gradient. Pour $k = 1, 2, \ldots, nr$ la composante k du vecteur gradient est

$$\frac{\partial J}{\partial \alpha_k} = 2 \int_S \left[T_{opt}(x) - T_0(x) - \sum_{k'=1}^{nr} \alpha_{k'} \, T_{k'}(x)\right] T_k(x) dx.$$

Définissons alors la matrice $\tilde{A} \in \mathbb{R}^{nr\times nr}$ et le vecteur $\tilde{b} \in \mathbb{R}^{nr}$ par

$$\tilde{A}_{k,k'} = \int_S T_k(x)\, T_{k'}(x)dx \quad \text{et} \quad \tilde{b}_k = \int_S T_k(x)\big(T_{opt}(x) - T_0(x)\big)\,dx.$$

La valeur optimale des résistances est obtenue comme solution du système linéaire

$$\tilde{A}\alpha = \tilde{b}. \tag{11.12}$$

11.6 RÉSOLUTION DU PROBLÈME INVERSE

Exercice 11.2

1. Construire et résoudre le système linéaire associé au problème d'optimisation de la température de cuisson.
2. Calculer le champ de température défini par les valeurs calculées des résistances. Discuter les résultats.

La solution de cet exercice est présentée à la page 244.

Dans un premier temps il faut résoudre les $nr+1$ problèmes directs qui permettent de calculer les champs de températures T_0 et T_k ($k = 1, 2, \ldots, nr$). On utilisera pour cela $nr + 1$ fois la procédure mise au point pour résoudre le problème direct, en modifiant les données T_D, f et F pour chaque calcul (il faut de plus choisir la position de chacune des résistances dans le four). Les champs de températures ainsi obtenus seront stockés dans $n + 1$ tableaux distincts.

Plus précisément, on résout d'abord un problème (11.1-11.2) avec un terme source nul ($F = 0$), mais avec une température $T_D \neq 0$ et flux de chaleur $f = 0$ imposés à la frontière. La solution de ce premier problème, notée T_0, est représentée sur la figure 11.4. On constate que les conditions aux limites sont bien respectées : température imposée $T = 100$ en $y = -1$ et $T = 50$ en $y = 1$. La conductivité thermique est $c = 1$ dans l'air et $c = 10$ dans l'objet. La condition de flux nul sur les deux autres côtés se traduit par des lignes isothermes perpendiculaires aux plans $x = -1$ et $x = 1$.

Ensuite on résout successivement autant de problèmes (11.1-11.2) qu'il y a de sources de chaleur (un problème par résistance). Chaque cas consiste à calculer la température du four quand une seule résistance fonctionne, ce qui revient à faire $F = 0$ dans chaque triangle de $\mathcal{T}_h$, sauf dans l'unique triangle qui contient la résistance, dans lequel on prend $F = 1$. Les conditions aux limites associées sont : température $T_D = 0$ sur $\partial\Omega_D$ et flux de chaleur $f = 0$ sur $\partial\Omega_N$. La température associée à la résistance k est notée T_k. Sur la figure 11.6 on voit une représentation d'un champ de température associé à une seule résistance. Noter la faible valeur du maximum ! On constate à nouveau que les conditions aux limites sont respectées : la

température est nulle en $y = -1$ et $y = 1$, la condition de flux nul se traduit par des lignes isothermes perpendiculaires aux plans verticaux $x = -1$ et $x = 1$.

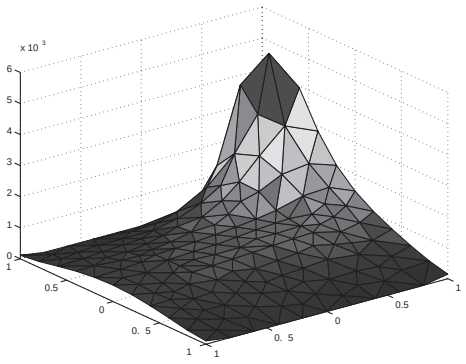


Figure 11.6 La température T_2

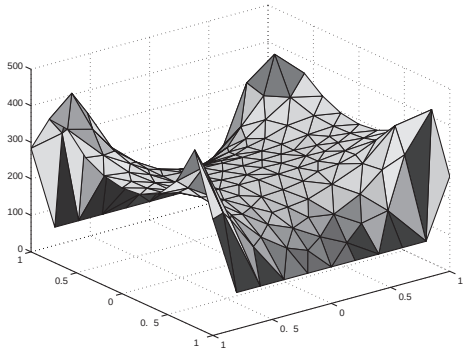


Figure 11.7 La température optimisée

Pour résoudre le problème inverse, il faut maintenant construire le système linéaire (11.12). Pour cela il faut calculer des intégrales du type

$$\int_S T_k(x)\ T_{k'}(x)dx$$

dans lesquelles les champs de température T_k et $T_{k'}$ sont connus. Pour effectuer ce calcul on utilise la même technique que pour le problème direct : on écrit d'abord que l'intégrale sur l'objet est la somme des intégrales sur chacun des triangles contenus dans l'objet

$$\int_S T_k(x)\ T_{k'}(x)dx = \sum_{K \subset S} \int_K T_k(x)\ T_{k'}(x)dx.$$

On évalue ensuite chacune des intégrales sur K, en reprenant l'expression particulière de $T_k(x)$ et $T_{k'}(x)$ dans le triangle

$$T_k(x) = T_k(A_3) + [T_k(A_1) - T_k(A_3)]\lambda_1 + [T_k(A_2) - T_k(A_3)]\lambda_2.$$

Dans cette formule λ_i est la $i^{ième}$ coordonnée barycentrique du point M de coordonnées x dans le triangle K de sommets A_i. On voit donc que l'on peut écrire

$$\int_K T_k(x)\ T_{k'}(x)dx = [dl\ T_{k,K}]^T\ [M_K]\ [dlT_{k',K}].$$

Ce qui introduit $[M_K]$, la matrice de masse de l'élément K. La surface de l'élément K est notée mes K. La matrice de masse du triangle de Lagrange de degré 1 est

$$[M_K] = \frac{1}{6} \text{mes}\, K \begin{bmatrix} 2 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 2 \end{bmatrix}.$$

Le coefficient $\tilde{A}_{k,k'}$ de la matrice du système linéaire (11.12) est obtenu par sommation sur tous les triangles internes à l'objet à cuire des quantités

$$[dl\ T_{k,K}]^T\ [M_K][dlT_{k',K}]$$

qui sont calculables puisque l'on connait les valeurs $T_{k,K}$ du champ de température $T_{k,K}$ aux trois sommets du triangle K. On procède de même pour calculer le second membre $\tilde{b}$. Un exemple de calcul - pour un système chauffant de 4 résistances - est présenté sur la figure 11.8. On y voit le champ de température optimisé obtenu après calcul des coefficients α_k. La figure 11.9 montre la solution obtenue pour un système chauffant de 6 résistances. On constate que la température sur l'objet S, contenu dans le rectangle $[-0.5, 0.5] \times [-0.2, 0.2]$ est proche de la température idéale de cuisson fixée à 250 degrés.

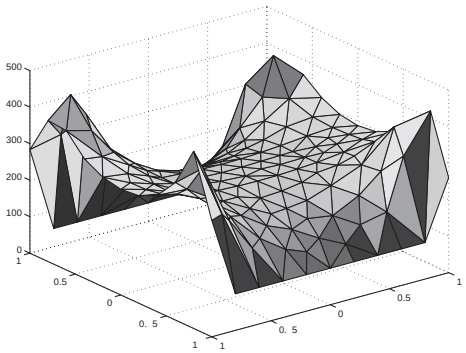


Figure 11.8 La température optimisée pour 4 résistances

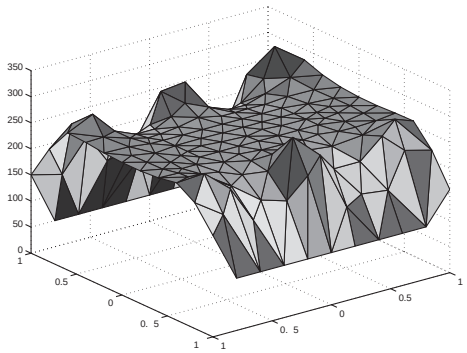


Figure 11.9 La température optimisée pour 6 résistances

La valeur optimale des coefficients α_k dépend du nombre et de l'emplacement des résistances par rapport à l'objet. Un prolongement intéressant de cette étude est d'optimiser la distribution géométrique des résistances dans le four. Le but pratique de cette recherche peut être d'optimiser la puissance thermique dissipée par les résistances pour obtenir la température idéale de cuisson (on veut en plus économiser l'énergie de chauffage). On peut modéliser cette puissance thermique par un terme supplémentaire dans la fonctionnelle que l'on minimise

$$J(\alpha) = \int_S \Big[T_{opt}(x) - T_0(x) - \sum_{k=1}^{nr} \alpha_k\ T_k(x)\Big]^2 dx + C\|\alpha\|_2^2.$$

Remarque 11.2 : Suivant le cas, on peut obtenir des valeurs α_k très petites ou même négatives. Ceci montre que les résistances correspondantes sont placées trop près de l'objet et doivent refroidir la pièce plutôt que la chauffer.

Sur la figure 11.10 on voit que le dispositif avec 6 résistances permet d'avoir une zone à température constante plus étendue, ce qui facilite la régulation thermique. Ces premiers calculs, effectués sur un maillage comprenant 173 sommets pour 304 triangles, donnent des résultats satisfaisants. Ils permettent de valider la démarche de modélisation ainsi que les procédures. Néanmoins, pour une approche plus réaliste et plus précise, il est nécessaire de résoudre le problème sur un maillage avec plus de points. Un second calcul est donc effectué sur un maillage comprenant 732 sommets pour 1 392 triangles et 6 résistances. Le résultat final est présenté sur la figure 11.11. On voit une nette amélioration de la représentation de la température au voisinage de l'objet et des résistances, due à un meilleure précision des calculs. Cette amélioration, cohérente avec la théorie de la méthode des éléments finis, se paie par une augmentation sensible du temps calcul.

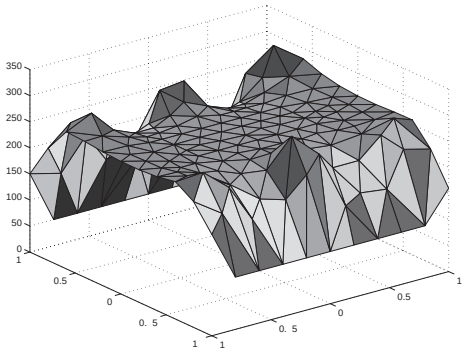


Figure 11.10 La température optimisée sur le maillage grossier

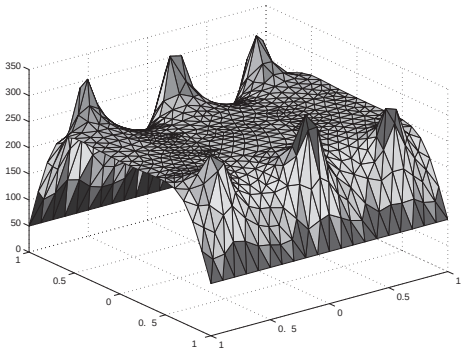


Figure 11.11 La température optimisée sur le maillage fin

11.7 SOLUTIONS ET PROGRAMMES

Toutes les procédures nécessaires à la résolution des exercices de ce chapitre peuvent être téléchargées à partir de la page web du livre.

Solution de l'exercice 11.1

La procédure du fichier `fourexo1.m` réalise l'expérimentation numérique proposée. Elle fixe les valeurs des paramètres physiques (localisation des résistances, conductivité des matériaux, valeur de la température au bord du four). Elle appelle la procédure du fichier `four.m` qui calcule les champs de température associés à ces données.

Listing 11.1 *four.m*

```
    function [T0,Tres,Numtri,Coorneu]=four(Refdir,Valdir,Conduc,
                                           Pres,Nomfich);
%-> Lecture du maillage
    [Nbpt,Nbtri,Coorneu,Refneu,Numtri,Reftri]=lecmail(Nomfich);
%-> Assemblage du système linéaire
    [Atotal]=assa(Nbpt,Nbtri,Coorneu,Refneu,Numtri,Reftri,Conduc);
%-> Les conditions aux limites
    [Aelim,belim]=elim(Atotal,Refneu,Refdir,Valdir);
%-> Résolution du problème sans source de chaleur
    T0=Aelim\belim;
    fprintf('\n Calcul de la temperature homogène')
%-> Résolution des problèmes avec source de chaleur
    for i=1:size(Pres,1),
      Pos=Pres(i,:);
      b=source(Pos,Numtri,Coorneu);
      Tres(:,i)=Aelim\b;
      fprintf('\n Calcul de la température associée à la résistance
%i',i);
    end
```

La procédure du fichier `assa.m` construit le système linéaire associé à l'équation de la chaleur. On note que le second membre est nul est dehors des éléments qui contiennent une résistance thermique. La procédure du fichier `local.m` construit le second membre à partir des coordonnées connues de chaque résistance. La procédure `elim.m` assure la prise en compte des conditions aux limites.

Listing 11.2 *assa.m*

```
   function [Atotal]=assa(Nbpt,Nbtri,Coorneu,Refneu, ...
                          Numtri,Reftri,Conduc);
    Atotal=sparse(Nbpt,Nbpt);
%-> Boucle sur les éléments
    for k=1:Nbtri
%--> Les coordonnées des sommets
       M=[Coorneu(Numtri(k,1),:); Coorneu(Numtri(k,2),:); ...
          Coorneu(Numtri(k,3),:)]';
%--> Le determinant
       Delta=abs(det([M; 1 1 1]));
%--> La matrice du gradient
       Dp=[M(2,2)-M(2,3) M(2,3)-M(2,1) M(2,1)-M(2,2);
       M(1,3)-M(1,2) M(1,1)-M(1,3) M(1,2)-M(1,1)];
       Dp=Dp/Delta;
```

```
%--> Assemblage de la matrice globale
       for i=1:3
       for j=1:3
          Atotal(Numtri(k,i),Numtri(k,j))=Atotal(Numtri(k,i),
                Numtri(k,j))+ ...
                Conduc(Reftri(k))*Delta*(Dp(:,i)'*Dp(:,j))/2;
       end
       end
   end
```

Solution de l'exercice 11.2

La procédure du fichier `fourexo2.m` réalise le calcul des valeurs résistances pour obtenir la température optimale de cuisson. Elle appelle la procédure du fichier `asst.m` qui calcule la matrice et le second membre du système linéaire associé au problème d'optimisation. Après résolution de ce système, le champ de température est obtenu par combinaison linéaire.

Listing 11.3 *four.m*

```
   function [Aopt,bopt]=asst(Nbpt,Nbtri,Coorneu,Refneu, ...
                             Numtri,Reftri,T0,Tres,Tcui);
    Nopt=size(Tres,2);
    Aopt=zeros(Nopt,Nopt);
    bopt=zeros(Nopt,1);
    Refcui = 1;
    for i=1:3
       for j=1:3
          AK(i,j)= 1/12.;
       end
       AK(i,i)= 1/6.;
    end
%-> Boucle sur les éléments
    for k=1:Nbtri
%--> Test sur le triangle
       if Reftri(k) == Refcui
%--> Les coordonnées des sommets
       M=[Coorneu(Numtri(k,1),:); Coorneu(Numtri(k,2),:);
                                          Coorneu(Numtri(k,3),:)]';
%--> Le déterminant
       Delta=abs(det([M; 1 1 1]));
%--> Assemblage de la matrice
       for i=1:Nopt
       for j=1:Nopt
```

```
         for ii=1:3
          for jj=1:3
             Aopt(i,j)=Aopt(i,j)+AK(ii,jj)*Delta*Tres(Numtri(k,ii),i)*
...
                          Tres(Numtri(k,jj),j);
          end
          end
       end
       end
%--> Assemblage du second membre
       for i=1:Nopt
          for ii=1:3
          for jj=1:3
             bopt(i)=bopt(i)+AK(ii,jj)*Delta*Tres(Numtri(k,ii),i)* ...
                     (Tcui-T0(Numtri(k,jj)));
          end
          end
       end
       end
   end
```

Remarque 11.3 : Il existe aussi une version interactive du projet qui permet de réaliser de nombreuses expériences numériques sans avoir à modifier les procédures.

BIBLIOGRAPHIE COMPLÉMENTAIRE OU LECTURE POUR APPROFONDIR

[Ciarlet-1978] P. G. Ciarlet *The finite element method for elliptic problems*, North Holland. Amsterdam (1978)

[Ciarlet-1982] P. G. Ciarlet *Introduction à l'analyse numérique matricielle et à l'optimisation*, Masson. Paris (1982)

[Joly] P. Joly *Analyse numérique matricielle*, Cassini. Paris (2004)

[Lucquin-Pironneau] B. Lucquin et O. Pironneau *Introduction au calcul scientifique*, Masson. Paris (1990)

[Mohammadi-Saiac] B. Mohammadi et J.-H. Saiac *Pratique de la simulation numérique*, Eyrolles. Paris (2003)

[Danaila-Hecht-Pironneau] I. Danaila, F. Hecht et O. Pironneau, *Simulation numérique en C++*, Dunod. Paris(2003)

Projet **12**

Mécanique des fluides : résolution des équations de Navier-Stokes 2D

Fiche Projet

Difficulté :	3
Notions développées :	Équations de Navier-Stokes, équation de Helmholtz, méthode de projection, factorisation ADI, transformée de Fourier rapide FFT
Domaines d'application :	Écoulements fluides 2D, instabilité de Kelvin-Helmholtz, dipôle de vorticité

Nous nous intéressons dans ce projet à la résolution des équations qui régissent la dynamique des fluides incompressibles (*i.e.* à masse volumique constante), les équations de Navier-Stokes.

Pour rendre ce problème (généralement assez difficile à résoudre) plus abordable, nous nous plaçons dans le cadre simplifié des équations bidimensionnelles (2D),

avec des conditions aux limites périodiques. Ce modèle nous permettra néanmoins de simuler quelques écoulements intéressants :

– l'instabilité de Kelvin-Helmholtz d'une couche de mélange plane,

– l'évolution d'un structure tourbillonnaire particulière, le dipôle de vorticité.

Sur le plan numérique, plusieurs algorithmes (d'intérêt beaucoup plus général) seront abordés :

– la discrétisation des équations par différences finies (schémas compacts à l'ordre 6) ;

– les schémas d'intégration en temps d'Adams-Bashfort et Crank-Nicolson ;

– la résolution d'une équation de Helmholtz par une méthode de factorisation approximative (ou ADI) ;

– la résolution d'un laplacien en utilisant une transformée de Fourier rapide (FFT) ;

– la résolution d'un système tridiagonal périodique par la méthode de Thomas.

12.1 ÉQUATIONS DE NAVIER-STOKES 2D, INCOMPRESSIBLES

L'écoulement 2D (dans le plan (x, y)) d'un fluide de masse volumique ρ constante est complètement décrit par son champ vectoriel de vitesse $q = (u(x, y), v(x, y)) \in \mathbb{R}^2$ et son champ scalaire de pression $p(x, y) \in \mathbb{R}$. Ces grandeurs seront déterminées à partir des lois de conservation suivantes (voir, par exemple, [Hirsh, 1988]) :

– la conservation de la masse :

$$div(q) = 0, \tag{12.1}$$

ou, en explicitant l'opérateur divergence[1] :

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0 \; ; \tag{12.2}$$

– la conservation de la quantité de mouvement, écrite sous forme compacte :

$$\frac{\partial q}{\partial t} + div(q \otimes q) = -\mathcal{G}p + \frac{1}{Re}\Delta q, \tag{12.3}$$

1. Rappelons les définitions sur $\mathbb{R}^2$ des opérateurs divergence, gradient et laplacien :

$$\text{si} \quad v = (v_x, v_y) : \mathbb{R}^2 \mapsto \mathbb{R}^2 \quad \text{et} \quad \varphi : \mathbb{R}^2 \mapsto \mathbb{R}$$

$$div(v) = \frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y}, \quad \mathcal{G}\varphi = (\frac{\partial \varphi}{\partial x}, \frac{\partial \varphi}{\partial y}), \quad \Delta\varphi = div(\mathcal{G}\varphi) = \frac{\partial^2 \varphi}{\partial x^2} + \frac{\partial^2 \varphi}{\partial y^2}, \quad \Delta v = (\Delta v_x, \Delta v_y).$$

ou, explicitée :

$$\left\{\begin{array}{rcl} \dfrac{\partial u}{\partial t}+\dfrac{\partial u^2}{\partial x}+\dfrac{\partial uv}{\partial y} & = & -\dfrac{\partial p}{\partial x}+\dfrac{1}{Re}\left(\dfrac{\partial^2 u}{\partial x^2}+\dfrac{\partial^2 u}{\partial y^2}\right) \\ \dfrac{\partial v}{\partial t}+\dfrac{\partial uv}{\partial x}+\dfrac{\partial v^2}{\partial y} & = & -\dfrac{\partial p}{\partial y}+\dfrac{1}{Re}\left(\dfrac{\partial^2 v}{\partial x^2}+\dfrac{\partial^2 v}{\partial y^2}\right). \end{array}\right. \tag{12.4}$$

Dans les équations précédentes, les variables s'entendent sans dimension, c'est-à-dire

$$x=\frac{x^*}{L},\quad y=\frac{y^*}{L},\quad u=\frac{u^*}{V_0},\quad v=\frac{v^*}{V_0},\quad t=\frac{t^*}{L/V_0},\quad p=\frac{p^*}{\rho V_0^2}, \tag{12.5}$$

où les variables $(^*)$ sont les grandeurs physiques et L, V_0 sont, respectivement, les échelles de longueur et de vitesse caractéristiques de l'écoulement. Le paramètre de similitude de l'écoulement est le nombre de Reynolds, défini comme le rapport entre les effets d'inertie et de viscosité dans le fluide :

$$Re=\frac{V_0 L}{\nu},\quad \text{avec } \nu \text{ la viscosité cinématique du fluide.} \tag{12.6}$$

En conclusion, le système d'équations de Navier-Stokes à résoudre est constitué par les équations aux dérivées partielles (EDP) (12.2), (12.4) ; la condition initiale (à $t = 0$) et les conditions aux limites seront précisées plus tard.

12.2 MÉTHODE DE RÉSOLUTION

Le système d'équations de Navier-Stokes sera résolu par une *méthode de projection* [Orlandi, 1999] comportant deux pas.

1. *Un pas de prédiction* : on résout les équations de quantité de mouvement (12.4) écrites sous la forme :

$$\frac{\partial q}{\partial t}=-\mathcal{G}p+\mathcal{H}+\frac{1}{Re}\Delta q,\quad \text{pour}\quad q=(u,v)\in\mathbb{R}^2, \tag{12.7}$$

avec le vecteur contenant les termes convectifs

$$-\mathcal{H}=\left(\frac{\partial u^2}{\partial x}+\frac{\partial uv}{\partial y},\ \frac{\partial uv}{\partial x}+\frac{\partial v^2}{\partial y}\right), \tag{12.8}$$

et le vecteur gradient de pression :

$$\mathcal{G}p=\left(\frac{\partial p}{\partial x},\frac{\partial p}{\partial y}\right). \tag{12.9}$$

Les équations (12.7) seront discrétisées suivant un schéma explicite d'Adams-Bashfort (pour les termes convectifs $\mathcal{H}$), combiné avec un schéma semi-implicite de Crank-Nicolson (pour les termes diffusifs Δq). L'avancement de la solution du temps $t_n = n\delta t$ au $t_{n+1} = (n+1)\delta t$ est donné par le schéma :

$$\frac{q^* - q^n}{\delta t} = -\mathcal{G}p^n + \underbrace{\frac{3}{2}\mathcal{H}^n - \frac{1}{2}\mathcal{H}^{n-1}}_{Adams-Bashfort} + \underbrace{\frac{1}{Re}\Delta\left(\frac{q^* + q^n}{2}\right)}_{Crank-Nicolson}. \tag{12.10}$$

Dans la formulation précédente, la pression apparaît en explicite (évaluée au temps t_n), ce qui fait que le champ de vitesse q^* ainsi calculé ne vérifie pas l'équation de continuité (12.1).

2. *Un pas de correction (ou projection)* : on corrige le champ q^* (dit *non-solénoïdal*) pour obtenir un champ de vitesse q^{n+1} (de divergence nulle, ou *solénoïdal*). L'équation de correction est la suivante[2] :

$$q^{n+1} - q^* = -\delta t\,\mathcal{G}\phi. \tag{12.11}$$

La variable ϕ (reliée à la pression, mais sans signification physique) sera déterminée en prenant la divergence de l'équation (12.11) ; tenant compte que $div(q^{n+1}) = 0$ et que $div(\mathcal{G}\phi) = \Delta\phi$, nous obtenons :

$$\Delta\phi = \frac{1}{\delta t} div(q^*). \tag{12.12}$$

Pour finir, l'algorithme est bouclé en réactualisant, pour le pas de temps suivant, la pression par[3]

$$p^{n+1} = p^n + \phi - \frac{\delta t}{2Re}\Delta\phi. \tag{12.13}$$

Pour résumer, l'algorithme de résolution comportera les étapes suivantes (en reprenant les équations sous une forme plus pratique pour la programmation) :

---

2. Cette équation exprime le fait que les champs q^* et q^{n+1} ont le même rotationnel (effectivement, on peut éliminer la pression des équations de Navier-Stokes en prenant le rotationnel des équations de quantité de mouvement).

3. Cette équation est obtenue en écrivant l'équation (12.10) avec la pression en implicite

$$\frac{q^{n+1} - q^n}{\delta t} = -\mathcal{G}p^{n+1} + \frac{3}{2}\mathcal{H}^n - \frac{1}{2}\mathcal{H}^{n-1} + \frac{1}{Re}\Delta\left(\frac{q^{n+1} + q^n}{2}\right)$$

et en faisant la différence avec (12.10). Après remplacement de q^* de l'équation de correction (12.11), nous obtenons (12.13), à une constante additive près. Observons que cette constante n'a aucune influence sur le déroulement de l'algorithme, car la pression intervient seulement par son gradient dans les équations de quantité de mouvement.

Algorithme 12.1. *Résolution des équations de Navier-Stokes (12.2)-(12.4) : à partir du champ* (u^n, v^n, p^n)*, il faut calculer*

(A) les termes explicites $\mathcal{H}^n$ *:*

$$\mathcal{H}_u^n = -\left(\frac{\partial u^2}{\partial x} + \frac{\partial uv}{\partial y}\right), \tag{12.14}$$

$$\mathcal{H}_v^n = -\left(\frac{\partial uv}{\partial x} + \frac{\partial v^2}{\partial y}\right), \tag{12.15}$$

(B) le champ non-solénoïdal $q^* = (u^*, v^*)$ *par la résolution des équations :*

$$\left(I - \frac{\delta t}{2Re}\Delta\right) u^* = u^n + \delta t\left[-\frac{\partial p^n}{\partial x} + \frac{3}{2}\mathcal{H}_u^n - \frac{1}{2}\mathcal{H}_u^{n-1} + \frac{1}{2Re}\Delta u^n\right], \tag{12.16}$$

$$\left(I - \frac{\delta t}{2Re}\Delta\right) v^* = v^n + \delta t\left[-\frac{\partial p^n}{\partial y} + \frac{3}{2}\mathcal{H}_v^n - \frac{1}{2}\mathcal{H}_v^{n-1} + \frac{1}{2Re}\Delta v^n\right], \tag{12.17}$$

(C) la variable ϕ *par la résolution de l'équation de Poisson :*

$$\Delta\phi = \frac{1}{\delta t}\left(\frac{\partial u^*}{\partial x} + \frac{\partial v^*}{\partial y}\right), \tag{12.18}$$

(D) le champ solénoïdal $q^{n+1} = (u^{n+1}, v^{n+1})$ *:*

$$u^{n+1} = u^* - \delta t\,\frac{\partial \phi}{\partial x}, \tag{12.19}$$

$$v^{n+1} = v^* - \delta t\,\frac{\partial \phi}{\partial y}, \tag{12.20}$$

(E) le nouveau champ de pression :

$$p^{n+1} = p^n + \phi - \frac{\delta t}{2Re}\Delta\phi. \tag{12.21}$$

Les étapes (A)-(E) seront reprises pour chaque pas de temps.

12.3 DOMAINE DE CALCUL, CONDITIONS AUX LIMITES ET MAILLAGE

Le domaine de calcul est rectangulaire, de dimensions $L_x \times L_y$ (voir fig. 12.1). Quant aux conditions aux limites, les conditions de périodicité simplifient considérablement la résolution des équations de Navier-Stokes. Nous considérons donc que les champs de vitesse $q(x, y)$ et de pression $p(x, y)$ sont périodiques en x et y :

$$q(0, y) = q(L_x, y), \qquad p(0, y) = p(L_x, y), \qquad \forall y, \tag{12.22}$$

$$q(x, 0) = q(x, L_y), \qquad p(x, 0) = p(x, L_y), \qquad \forall x. \tag{12.23}$$

Les équations seront discrétisées sur un maillage de type différences finies en utilisant n_x points de discrétisation suivant la direction x et n_y points suivant la direction y. Le maillage *primaire* est défini par les points de coordonnées (voir fig. 12.1)

$$x_c(i) = (i-1)\delta x, \qquad \delta x = \frac{L_x}{n_x - 1}, \qquad i = 1, \ldots, n_x \tag{12.24}$$

$$y_c(j) = (j-1)\delta y, \qquad \delta y = \frac{L_y}{n_y - 1}, \qquad j = 1, \ldots, n_y. \tag{12.25}$$

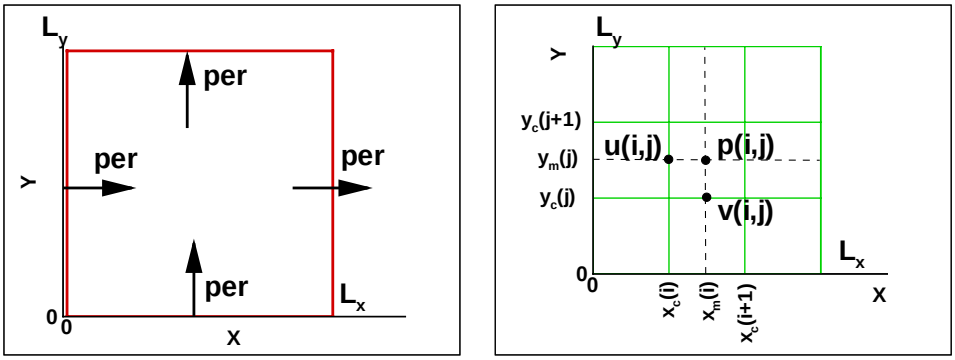


Figure 12.1 Domaine de calcul, conditions aux limites et maillage décalé.

Les centres des mailles forment le maillage *secondaire*, défini par les coordonnées :

$$x_m(i) = (i - 1/2)\delta x, \qquad i = 1, \ldots, n_{xm} \tag{12.26}$$

$$y_m(j) = (j - 1/2)\delta y, \qquad j = 1, \ldots, n_{ym}, \tag{12.27}$$

où, pour simplifier la présentation, on a noté : $n_{xm} = n_x - 1$, $n_{ym} = n_y - 1$.

Dans la maille (i, j), *i.e.* le rectangle $[x_c(i), x_c(i+1)] \times [y_c(j), y_c(j+1)]$, les inconnues u, v, p seront évaluées en des points distincts :

- $u(i, j)$ est calculé au point $(x_c(i), y_m(j))$ (face ouest),
- $v(i, j)$ au point $(x_m(i), y_c(j))$ (face sud),
- et $p(i, j)$ au point $(x_m(i), y_m(j))$ (centre de la maille).

On parle dans ce cas d'une discrétisation sur un maillage *décalé*, cette technique assurant une meilleure stabilité de la méthode numérique décrite précédemment.

12.4 DISCRÉTISATION DES ÉQUATIONS

Dans ce paragraphe, l'algorithme 12.1 (page 250) sera écrit sous sa forme discrète, qui sera ensuite programmée. Commençons par observer que les conditions aux limites de périodicité prennent la forme discrète suivante :

$$\begin{aligned} u(1, j) &= u(n_x, j), \quad \forall j = 1, \ldots n_y \\ u(i, 1) &= u(i, n_y), \quad \forall i = 1, \ldots n_x, \end{aligned} \tag{12.28}$$

plus les relations similaires pour v et p. Les inconnues du problème seront, par conséquent, les valeurs discrètes :

$$u(i,j), \quad v(i,j), \quad p(i,j), \quad \text{pour} \quad i = 1,\ldots n_{xm}, \quad j = 1,\ldots n_{ym}.$$

Une astuce de programmation pour prendre en compte facilement les conditions aux limites (12.28) dans la discrétisation des opérateurs différentiels est d'introduire les vecteurs suivants :

$$\begin{cases} ip(i) = i+1, \quad i = 1,\ldots(n_{xm}-1) \\ ip(n_{xm}) = 1 \end{cases}, \quad \begin{cases} jp(j) = j+1, \quad j = 1,\ldots(n_{ym}-1) \\ jp(n_{ym}) = 1 \end{cases} \tag{12.29}$$

$$\begin{cases} im(i) = i-1, \quad i = 2,\ldots n_{xm} \\ im(1) = n_{xm} \end{cases}, \quad \begin{cases} jm(j) = j-1, \quad j = 2,\ldots n_{ym} \\ jm(1) = n_{ym} \end{cases} \tag{12.30}$$

Avec ces notations, on peut écrire les schémas aux différences finies de manière plus compacte et finalement utiliser ces expressions dans la programmation, grâce aux capacités de calcul vectoriel de MATLAB . Par exemple, pour calculer $\partial\psi/\partial x(i,j)$ pour j fixé et $i = 1,\ldots,n_{xm}$, le schéma aux différences finies centrées à l'ordre deux s'écrit

$$\frac{\partial\psi}{\partial x}(i,j) \simeq \frac{\psi(i+1,j)-\psi(i-1,j)}{2\delta x}, \qquad i = 2\ldots(n_{xm}-1),$$

avec un traitement particulier pour $i = 1$ et $i = n_{xm}$:

$$\frac{\partial\psi}{\partial x}(1,j) \simeq \frac{\psi(2,j)-\psi(n_{xm},j)}{2\delta x}, \quad \frac{\partial\psi}{\partial x}(n_{xm},j) \simeq \frac{\psi(1,j)-\psi(n_{xm}-1,j)}{2\delta x}.$$

En introduisant les vecteurs *im* et *ip* (12.29, 12.30), on peut écrire directement :

$$\frac{\partial\psi}{\partial x}(i,j) \simeq \frac{\psi(ip(i),j)-\psi(im(i),j)}{2\delta x}, \quad i = 1,\ldots,n_{xm}. \tag{12.31}$$

Comme règle générale, dans les schémas aux différences finies avec des conditions de périodicité, on va remplacer les indices $(i+1)$ par $ip(i)$ et $(i-1)$ par $im(i)$ (de même pour j).

Regardons maintenant la forme discrète de chaque étape de l'algorithme 12.1.

a) Calcul des termes explicites (A)

Les deux composantes $\mathcal{H}_u^n$ et $\mathcal{H}_v^n$ du terme explicite $\mathcal{H}^n$ (équations 12.14, 12.15) seront calculées aux mêmes points que les composantes de vitesse correspondantes. En utilisant le schéma aux différences finies centrées (12.31), on obtient les discrétisations suivantes (à suivre sur la figure 12.1) :

- pour le calcul de la vitesse u (position $(x_c(i), y_m(j))$) : pour $i=1,\ldots n_{xm}, j=1,\ldots n_{ym}$

$$\frac{\partial u^2}{\partial x}(i,j) \simeq \frac{1}{\delta x}\left[\left(\frac{u(i,j)+u(ip(i),j)}{2}\right)^2-\left(\frac{u(i,j)+u(im(i),j)}{2}\right)^2\right] \quad (12.32)$$

$$\frac{\partial uv}{\partial y}(i,j) \simeq \frac{1}{\delta y}\left[\left(\frac{u(i,j)+u(i,jp(j))}{2}\right)\left(\frac{v(i,jp(j))+v(im(i),jp(j))}{2}\right)\right. \quad (12.33)$$
$$\left.-\left(\frac{u(i,j)+u(i,jm(j))}{2}\right)\left(\frac{v(i,j)+v(im(i),j)}{2}\right)\right]$$

$$\mathcal{H}_u^n(i,j) = -\frac{\partial u^2}{\partial x}(i,j)-\frac{\partial uv}{\partial y}(i,j), \quad (12.34)$$

- et de manière similaire pour la vitesse v (position $(x_m(i), y_c(j))$) : pour $i=1,\ldots n_{xm}$, $j=1,\ldots n_{ym}$

$$\frac{\partial v^2}{\partial y}(i,j) \simeq \frac{1}{\delta y}\left[\left(\frac{v(i,j)+v(i,jp(j))}{2}\right)^2-\left(\frac{v(i,j)+u(i,jm(j))}{2}\right)^2\right] \quad (12.35)$$

$$\frac{\partial uv}{\partial x}(i,j) \simeq \frac{1}{\delta x}\left[\left(\frac{u(ip(i),j)+u(ip(i),jm(j))}{2}\right)\left(\frac{v(i,j)+v(ip(i),j)}{2}\right)\right. \quad (12.36)$$
$$\left.-\left(\frac{u(i,j)+u(i,jm(j))}{2}\right)\left(\frac{v(i,j)+v(im(i),j)}{2}\right)\right]$$

$$\mathcal{H}_v^n(i,j) = -\frac{\partial uv}{\partial x}(i,j)-\frac{\partial v^2}{\partial y}(i,j). \quad (12.37)$$

b) Calcul du champ non-solénoïdal (B)

Observons que les équations (12.16) et (12.17) peuvent s'écrire sous la forme d'une équation de Helmholtz :

$$\underbrace{\left(I-\frac{\delta t}{2Re}\Delta\right)}_{\text{opérateur de Helmholtz}}\delta q^* = \underbrace{\delta t\left[-\mathcal{G}p^n+\frac{3}{2}\mathcal{H}^n-\frac{1}{2}\mathcal{H}^{n-1}+\frac{1}{Re}\Delta q^n\right]}_{RHS^n}, \quad (12.38)$$

où la notation $\delta q^* = q^* - q^n$ a été introduite. Pour résoudre cette équation, plusieurs méthodes sont envisageables. Nous allons utiliser la plus simple à mettre en œuvre : la factorisation approximative (ou ADI - *Alternating Direction Implicit* en anglais). L'opérateur de Helmholtz est approché, avec une précision à l'ordre deux en temps, sous la forme :

$$\left(I-\frac{\delta t}{2Re}\Delta\right)\delta q^* \simeq \left(I-\frac{\delta t}{2Re}\frac{\partial^2}{\partial x^2}\right)\left(I-\frac{\delta t}{2Re}\frac{\partial^2}{\partial y^2}\right)\delta q^*, \quad (12.39)$$

où les termes en $\mathcal{O}(\delta t^2)$ ont été négligés. Cette factorisation est utilisée pour résoudre l'équation (12.38) en deux étapes :

$$\left(I - \frac{\delta t}{2Re}\frac{\partial^2}{\partial x^2}\right)\overline{\delta q^*} = RHS^n \quad (+ \quad \text{condition de périodicité en } x) \tag{12.40}$$

$$\left(I - \frac{\delta t}{2Re}\frac{\partial^2}{\partial y^2}\right)\delta q^* = \overline{\delta q^*} \quad (+ \quad \text{condition de périodicité en } y). \tag{12.41}$$

Observons que nous avons associé au champ $\overline{\delta q^*}$, qui n'a pas de signification physique, les mêmes conditions aux limites que pour δq^*. Ce choix, qui est naturel pour des conditions de périodicité, nécessite des développements plus poussés pour d'autres types de conditions aux limites (conditions de Dirichlet par exemple).

Les dérivées secondes intervenant dans les équations (12.40) et (12.41) seront approchées par des différences centrées à l'ordre deux, suivant le schéma général :

$$\begin{aligned}\frac{\partial^2\psi}{\partial x^2}(i,j) &\simeq \frac{\psi(i+1,j) - 2\psi(i,j) + \psi(i-1,j)}{\delta x^2}\\ \frac{\partial^2\psi}{\partial y^2}(i,j) &\simeq \frac{\psi(i,j+1) - 2\psi(i,j) + \psi(i,j-1)}{\delta y^2},\end{aligned} \tag{12.42}$$

ce qui conduit à l'algorithme suivant :

Algorithme 12.2. *Calcul de la vitesse u^* par l'algorithme ADI.*

- *Premier pas d'ADI. Pour chaque $j = 1, \ldots n_{ym}$ il faut résoudre le système suivant i :*

$$-\beta_x\overline{(\delta u^*)}(i-1,j) + (1+2\beta_x)\overline{(\delta u^*)}(i,j) - \beta_x\overline{(\delta u^*)}(i+1,j) = RHS_u^n(i,j), \tag{12.43}$$

avec $i = 1, \ldots n_{xm}$ et $\beta_x = \dfrac{\delta t}{2Re}\dfrac{1}{\delta x^2}$ et la convention (imposée par la condition de périodicité)

$$\overline{(\delta u^*)}(0,j) = \overline{(\delta u^*)}(n_{xm},j), \qquad \overline{(\delta u^*)}(n_{xm}+1,j) = \overline{(\delta u^*)}(1,j).$$

Il s'agit en fait de n_{ym} systèmes linéaires à matrice tridiagonale périodique de taille $n_{xm} \times n_{xm}$:

$$M_x = \begin{pmatrix} 1+2\beta_x & -\beta_x & 0 & . & . & 0 & 0 & -\beta_x \\ -\beta_x & 1+2\beta_x & -\beta_x & . & . & 0 & 0 & 0 \\ . & . & . & . & . & . & . & . \\ 0 & 0 & 0 & . & . & -\beta_x & 1+2\beta_x & -\beta_x \\ -\beta_x & 0 & 0 & . & . & 0 & -\beta_x & 1+2\beta_x \end{pmatrix} \tag{12.44}$$

Une méthode rapide qui permet de résoudre simultanément ces systèmes sera développée à partir de l'algorithme de Thomas dans les questions préliminaire de ce projet (page 262).
Nous obtenons à la fin de ce pas $\overline{(\delta u^*)}(i,j)$.

- *Deuxième pas d'ADI. Pour chaque* $i = 1, \ldots n_{xm}$ *il faut résoudre le système suivant* j *:*

$$-\beta_y(\delta u^*)(i,j-1)+(1+2\beta_y)(\delta u^*)(i,j)-\beta_y(\delta u^*)(i,j+1) = \overline{(\delta u^*)}(i,j), \quad (12.45)$$

avec $j = 1, \ldots n_{ym}$ *et* $\beta_y = \dfrac{\delta t}{2Re}\dfrac{1}{\delta y^2}$ *et la convention (imposée par la condition de périodicité)*

$$(\delta u^*)(i,0) = -(\delta u^*)(i,n_{xm}), \qquad (\delta u^*)(i,n_{ym}+1) = (\delta u^*)(i,1).$$

Les n_{xm} *systèmes linéaires ont le même type de matrice que précédemment - elle sera de taille* $n_{ym} \times n_{ym}$ *:*

$$M_y = \begin{pmatrix} 1+2\beta_y & -\beta_y & 0 & . & . & 0 & 0 & -\beta_y \\ -\beta_y & 1+2\beta_y & -\beta_y & . & . & 0 & 0 & 0 \\ . & . & . & . & . & . & . & . \\ 0 & 0 & 0 & . & . & -\beta_y & 1+2\beta_y & -\beta_y \\ -\beta_y & 0 & 0 & . & . & 0 & -\beta_y & 1+2\beta_y \end{pmatrix} \quad (12.46)$$

Après la résolution des systèmes, nous obtenons $(\delta u^*)(i,j)$ *et immédiatement*

$$u^*(i,j) = u(i,j) + (\delta u^*)(i,j).$$

La procédure de calcul sera tout à fait similaire pour l'autre composante de la vitesse :

Algorithme 12.3. *Calcul de la vitesse* v^* *par l'algorithme ADI.*

- *Premier pas d'ADI. Pour chaque* $j = 1, \ldots n_{ym}$ *il faut résoudre le système suivant* i *:*

$$M_x\,\overline{(\delta v^*)}(i,j) = RHS_v^n(i,j), \quad (12.47)$$

avec $i = 1, \ldots, n_{xm}$ *et la matrice* M_x *donnée par l'équation (12.44). Nous obtenons à la fin de ce pas* $\overline{(\delta v^*)}(i,j)$*.*

- *Deuxième pas d'ADI. Pour chaque* $i = 1, \ldots n_{xm}$*, il faut résoudre le système suivant* j *:*

$$M_y\,\overline{(\delta v^*)}(i,j) = \overline{(\delta v^*)}(i,j), \quad (12.48)$$

avec $j = 1, \ldots, n_{ym}$ *et la matrice* M_y *donnée par l'équation (12.46).*

Nous obtenons $(\delta v^*)(i,j)$ *et immédiatement*

$$v^*(i,j) = u(i,j) + (\delta v^*)(i,j).$$

c) Résolution de l'équation de Poisson (C)

L'équation de Poisson (12.12) s'écrit sous la forme discrète suivante :

$$\Delta\phi(i,j) = (\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2})\phi(i,j) = Q(i,j),\ \ i = 1,\ldots n_{xm},\ \ j = 1,\ldots n_{ym}, \qquad (12.49)$$

avec

$$Q(i,j) = \frac{1}{\delta t} div(q^*)(i,j) = \frac{1}{\delta t}\left(\frac{\partial u^*}{\partial x} + \frac{\partial v^*}{\partial y}\right)(i,j).$$

Pour résoudre cette équation, on va utiliser d'abord la périodicité de la variable ϕ suivant la direction x ; on peut donc utiliser une décomposition en série de Fourier

$$\phi(i,j) = \sum_{l=1}^{n_{xm}} \widehat{\phi}_l(j) e^{\mathbf{i}\frac{2\pi}{n_{xm}}(i-1)(l-1)}, \quad \forall\, i = 1\ldots n_{xm}, \qquad (12.50)$$

où $\mathbf{i} = \sqrt{-1}$ est l'unité imaginaire. L'utilité de cette décomposition est de diagonaliser l'opérateur laplacien et de ramener le problème initial (12.49) à un problème 1D. Effectivement, en utilisant une approximation de $\partial^2\phi(i,j)/\partial x^2$ par différences centrées, on obtient :

$$\frac{\partial^2}{\partial x^2}\phi(i,j) \simeq \frac{\phi(i+1,j) - 2\phi(i,j) + \phi(i-1,j)}{\Delta x^2} \qquad (12.51)$$

$$= \frac{1}{\Delta x^2}\sum_{l=1}^{n_{xm}} \widehat{\phi}_l(j) e^{\mathbf{i}\frac{2\pi}{n_{xm}}(i-1)(l-1)}\left(e^{\mathbf{i}\frac{2\pi}{n_{xm}}(l-1)} - 2 + e^{-\mathbf{i}\frac{2\pi}{n_{xm}}(l-1)}\right)$$

$$= \sum_{l=1}^{n_{xm}} \widehat{\phi}_l(j) e^{\mathbf{i}\frac{2\pi}{n_{xm}}(i-1)(l-1)} \underbrace{\frac{2}{\Delta x^2}\left[\cos\left(\frac{2\pi}{n_{xm}}(l-1)\right) - 1\right]}_{k_l}$$

Le second membre Q étant aussi périodique, il est également décomposé en série de Fourier,

$$Q(i,j) = \sum_{l=1}^{n_{xm}} \widehat{Q}_l(j) e^{\mathbf{i}\frac{2\pi}{n_{xm}}(i-1)(l-1)},$$

ce qui fait que l'équation initiale (12.49) est équivalente à la résolution, pour chaque *nombre d'onde* $l = 1,\ldots,n_{xm}$, d'une équation de la forme :

$$\frac{\partial^2}{\partial y^2}\widehat{\phi}_l(j) + k_l\widehat{\phi}_l(j) = \widehat{Q}_l(j). \qquad (12.52)$$

À partir de ce point, nous avons le choix entre deux méthodes : utiliser une décomposition de Fourier suivant y, ou discrétiser directement la dérivée seconde suivant y par des différences finies centrées.

Nous choisissons la deuxième méthode, qui a l'avantage de pouvoir être utilisée dans le cas d'une condition à la limite différente suivant y (condition de paroi par

exemple). L'équation (12.52) devient :

$$\frac{1}{\delta y^2}\widehat{\phi}_l(j-1) + (-\frac{2}{\delta y^2} + k_l)\widehat{\phi}_l(j) + \frac{1}{\delta y^2}\widehat{\phi}_l(j+1) = \widehat{Q}_l(j), \quad j = 1, \ldots, n_{ym}. \quad (12.53)$$

Pour cette dernière équation nous avons besoin de conditions aux limites pour $j = 1$ et $j = n_{ym}$. Dans notre cas, la condition de périodicité est naturellement imposée.

Un traitement particulier doit être réservé au nombre d'onde $l = 1$ pour lequel $k_l = 0$. Dans ce cas, il est facile de voir que la matrice du système (12.53) est singulière, ce qui traduit le fait que le problème est mal posé ! Effectivement, la condition de périodicité fait que la solution du laplacien est déterminée à une constante près : cette constante est donnée exactement par la valeur du premier coefficient de la série de Fourier (la valeur moyenne). Nous pouvons donc imposer librement cette constante sans aucune incidence sur les calculs, car, même si la variable ϕ va déterminer le champ de pression à une constante près, dans les équations de Navier-Stokes seulement le gradient de la pression intervient !

Il est donc naturel d'imposer pour $l = 1$, $\widehat{\phi}_l(j) = 0, j = 1, \ldots n_{ym}$ (moyenne nulle).

Algorithme 12.4. *En pratique, la résolution de l'équation de Poisson comportera les étapes suivantes (points de coordonnées* $(x_m(i), y_m(j))$*) :*

- *on calcule le tableau* $Q(i,j), i = 1, \ldots n_{xm}, \quad j = 1, \ldots n_{ym}$ *:*

$$Q(i,j) = \frac{1}{\delta t}\left(\frac{u^*(ip(i),j) - u^*(i,j)}{\delta x} + \frac{v^*(i,jp(i)) - v^*(i,j)}{\delta y}\right) ; \quad (12.54)$$

- *on effectue une transformée de Fourier rapide (FFT) de chaque colonne du tableau* Q

$$\widehat{Q}(l,j) = FFT(Q(i,j)), \quad l = 1, \ldots n_{xm}, \quad j = 1, \ldots n_{ym} \quad (12.55)$$

  (Attention, les valeurs $\widehat{Q}$ *sont complexes !) ;*

- *pour* $l = 1$ *on impose* $\widehat{\phi}_1(j) = 0, j = 1, \ldots n_{ym}$ *;*

- *pour chaque* $l = 2, \ldots n_{xm}$ *on résout le système* $M_l\widehat{\phi}_l = \widehat{Q}(l,j)^T$ *à matrice tridiagonale* $(n_{ym} \times n_{ym})$ *:*

$$M_l = \frac{1}{\delta y^2}\begin{bmatrix} -2+\delta y^2 k_l & 1 & 0 & . & . & 0 & 0 & 1 \\ 1 & -2+\delta y^2 k_l & 1 & . & . & 0 & 0 & \\ . & . & . & . & . & . & . & . \\ 0 & 0 & 0 & . & . & 1 & -2+\delta y^2 k_l & 1 \\ 1 & 0 & 0 & . & . & 0 & 1 & -2+\delta y^2 k_l \end{bmatrix}, \quad (12.56)$$

  avec

$$k_l = \frac{2}{\Delta x^2}\left[\cos\left(\frac{2\pi}{n_{xm}}(l-1)\right) - 1\right] ;$$

- *on construit le tableau* $\widehat{\Phi}(l,j)$*, dont les lignes sont les vecteurs* $\widehat{\phi}_l$ *;*

- *enfin, par une transformée de Fourier inverse (IFFT), on obtient la solution*

$$\phi(i,j) = IFFT(\widehat{\Phi}(l,j)), \qquad i = 1,\ldots n_{xm}, \quad j = 1,\ldots n_{ym}. \tag{12.57}$$

d) Calcul du champ solénoïdal (D)

Une fois le tableau $\phi(i,j)$ calculé, il est très facile de corriger le champ de vitesse :

- pour $i = 1,\ldots n_{xm}, \quad j = 1,\ldots n_{ym}$

$$u^{n+1}(i,j) = u^*(i,j) - \delta t\, \frac{\phi(i,j) - \phi(im(i),j)}{\Delta x},. \tag{12.58}$$

- pour $i = 1,\ldots n_{xm}, \quad j = 1,\ldots n_{ym}$

$$v^{n+1}(i,j) = v^*(i,j) - \delta t\, \frac{\phi(i,j) - \phi(i,jm(j))}{\Delta y}.. \tag{12.59}$$

e) Calcul du champ de pression (E)

Le calcul du nouveau champ de pression est immédiat suivant l'équation (12.21) :

- pour $i = 1,\ldots n_{xm}, \quad j = 1,\ldots n_{ym}$

$$p^{n+1}(i,j) = p^n(i,j) + \phi(i,j) - \frac{\delta t}{2Re}\left[\frac{\phi(ip(i),j) - 2\phi(i,j) + \phi(im(i),j)}{\delta x^2} + \frac{\phi(i,jp(j)) - 2\phi(i,j) + \phi(i,jm(j))}{\delta y^2}\right] \tag{12.60}$$

Le gradient de pression doit être également réactualisé :

- pour $i = 1,\ldots n_{xm}, \quad j = 1,\ldots n_{ym}$

$$\frac{\partial p^{n+1}}{\partial x}(i,j) = \frac{p^{n+1}(i,j) - p^{n+1}(im(i),j)}{\delta x}, \tag{12.61}$$

- pour $i = 1,\ldots n_{xm}, \quad j = 1,\ldots n_{ym}$

$$\frac{\partial p^{n+1}}{\partial y}(i,j) = \frac{p^{n+1}(i,j) - p^{n+1}(i,jm(j))}{\delta y}. \tag{12.62}$$

f) Calcul du pas de temps

Pour que l'algorithme de résolution soit complet, il faut indiquer comment calculer le pas de temps δt. Comme nous utilisons un schéma semi-implicite, la valeur du pas de temps sera limitée par une condition de type CFL (condition de stabilité du schéma, voir le Projet 1, page 1). La dérivation de cette condition par une analyse de stabilité étant assez complexe, la valeur du pas de temps sera calculée à partir de la

relation générale :

$$dt = \frac{cfl}{max\left(\left|\frac{u}{\delta x}\right| + \left|\frac{v}{\delta y}\right|\right)}, \tag{12.63}$$

avec la constante $cfl < 1$. Afin d'optimiser les différentes procédures de calcul, nous allons utiliser un pas de temps constant, évalué à partir de la condition initiale.

12.5 VISUALISATION DE L'ÉCOULEMENT

Nous disposons de deux moyens rapides de visualisation de l'évolution de l'écoulement. Le premier consiste à calculer le champ de vorticité, suivant la définition suivante :

$$\omega = \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y}, \tag{12.64}$$

avec les valeurs discrètes calculées aux points $(x_c(i), y_c(j))$ par :

$$\omega(i,j) = \frac{v(i,j) - v(im(i),j)}{\delta x} - \frac{u(i,j) - u(i,jm(j))}{\delta y}. \tag{12.65}$$

Les iso-valeurs (ou lignes de niveau) de la vorticité nous permettront d'identifier les tourbillons dans le champ de l'écoulement (les tourbillons étant définis comme concentrations de vorticité).[4]

Un autre moyen de visualisation est de suivre l'évolution d'un scalaire (traceur) passif dans l'écoulement. Le scalaire passif, comme son nom l'indique, est simplement transporté par le champ de vitesse sans influencer l'évolution de l'écoulement. Il est l'équivalent numérique du colorant utilisé dans les visualisations expérimentales.

L'évolution du scalaire passif χ est décrite par une équation de convection-diffusion :

$$\frac{\partial \chi}{\partial t} + \frac{\partial \chi u}{\partial x} + \frac{\partial \chi v}{\partial y} = \frac{1}{Pe}\Delta\chi, \tag{12.66}$$

où le paramètre de similitude Pe (le nombre de Peclet) caractérise les propriétés de diffusion du traceur passif. L'équation (12.66) sera résolue suivant exactement le même schéma (Adams-Bashfort + Crank-Nicolson) utilisé pour les équations de quantité de mouvement. Cette résolution interviendra après le calcul du champ solénoïdal.

Pour voir des images illustrant les structures tourbillonnaires présentes dans différents écoulements fluides réels, le lecteur peut consulter [Lesieur, 1994].

4. MATLAB fournit de fonctions qui tracent les iso-valeurs pour un champ 2D (fonctions `contour` ou `pcolor`).

12.6 CONDITION INITIALE

Nous avons vu comment avancer la solution en temps. Il nous reste à spécifier la condition initiale qui va préciser le type d'écoulement étudié. Le champ initial doit être compatible avec les équations de Navier-Stokes ; en pratique, il suffit de prescrire le champ dynamique (vitesses u et v) et garder une pression nulle partout. Le champ de pression correct s'établira rapidement au cours du calcul. Nous allons simuler deux types d'écoulements.

12.6.1 Dynamique d'un jet plan. Instabilité de Kelvin-Helmholtz

L'instabilité de Kelvin-Helmholtz se produit lorsqu'il existe dans le champ de l'écoulement un cisaillement entre les couches fluides. Une expérience très simple pour mettre en évidence cette instabilité consiste à disposer dans un long parallélépipède un mélange de deux liquides dont l'un est plus lourd que l'autre. Une simple inclinaison du parallélépipède provoque l'écoulement de la partie plus lourde vers le bas en repoussant la partie légère vers le haut. On observe une ondulation de la surface de séparation qui s'amplifie pour donner naissance à de beaux tourbillons, semblables à des vagues (voir figure 12.2).

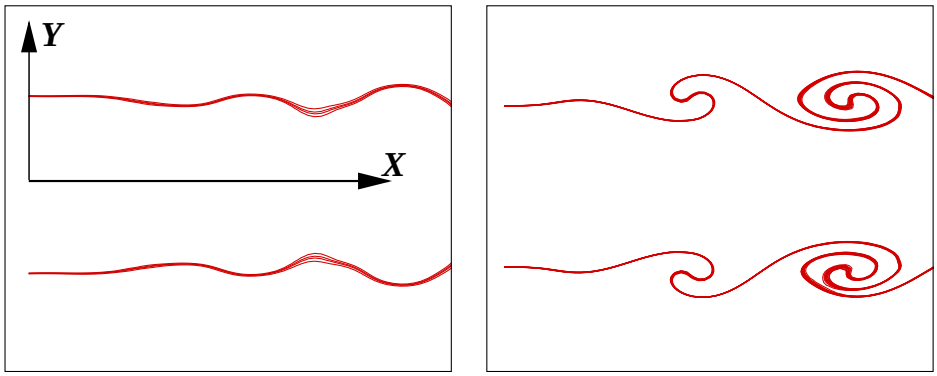


Figure 12.2 Représentation schématique du développement de l'instabilité de Kelvin-Helmholtz : perturbation de la couche de cisaillement (à gauche) et enroulement des tourbillons (à droite).

Pour observer numériquement ce phénomène, nous allons imposer un cisaillement dans le champ fluide en introduisant un profil de vitesse avec un point d'inflexion.

La condition initiale correspond à un écoulement de jet plan :

$$v(x,y) = 0, \qquad u(x,y) = u_1(y)(1 + u_2(x))$$

avec le profil de base (P_j est le paramètre du jet et R_j le rayon du jet) :

$$u_1(y)/U_0 = \frac{1}{2}\left(1 + tanh\left(\frac{1}{2}P_j\left(1 - \frac{|L_y/2 - y|}{R_j}\right)\right)\right); \qquad (12.67)$$

et la perturbation (nécessaire pour engendrer l'instabilité de Kelvin-Helmholtz) :

$$u_2(x) = A_x \, sin\left(2\pi \frac{x}{\lambda_x}\right). \tag{12.68}$$

12.6.2 Mouvement d'un dipôle de vorticité

Un dipôle de vorticité est une paire de deux tourbillons de vorticités égales et de signes opposés (sens de rotation opposés). Comme chaque tourbillon induit dans le champ de l'écoulement une vitesse orientée suivant son sens de rotation et proportionnelle à son intensité, le dipôle sera une structure stable qui se déplace suivant son axe de symétrie avec une vitesse constante.

Nous allons construire un dipôle de vorticité en juxtaposant deux tourbillons individuels suivant un modèle très simple. Un tourbillon centré en (x_v, y_v), de taille l_v et intensité ψ_0, est décrit analytiquement par la fonction de courant :

$$\psi(x,y) = \psi_0 \, exp\left(-\frac{(x-x_v)^2 + (y-y_v)^2}{l_v^2}\right), \tag{12.69}$$

ce qui nous permet de calculer les deux composantes du vecteur vitesse par :

$$\begin{cases} u = \dfrac{\partial \psi}{\partial y} = -\,2\dfrac{(y-y_v)}{l_v^2}\,\psi(x,y) \\ v = -\dfrac{\partial \psi}{\partial x} = 2\dfrac{(x-x_v)}{l_v^2}\,\psi(x,y). \end{cases} \tag{12.70}$$

Un dipôle sera maintenant construit par la superposition des champs de vitesse de deux tourbillons de même taille et intensité, distribués symétriquement par rapport à une direction choisie. Par exemple, un dipôle se propageant suivant l'axe des x, peut être obtenu en superposant (voir fig. 12.3) deux tourbillons définis par :
tourbillon 1 : $+\psi_0, l_v, x_v, y_v = L_y + a$
tourbillon 2 : $-\psi_0, l_v, x_v, y_v = L_y - a$,
avec a une distance définissant la distance entre les tourbillons.

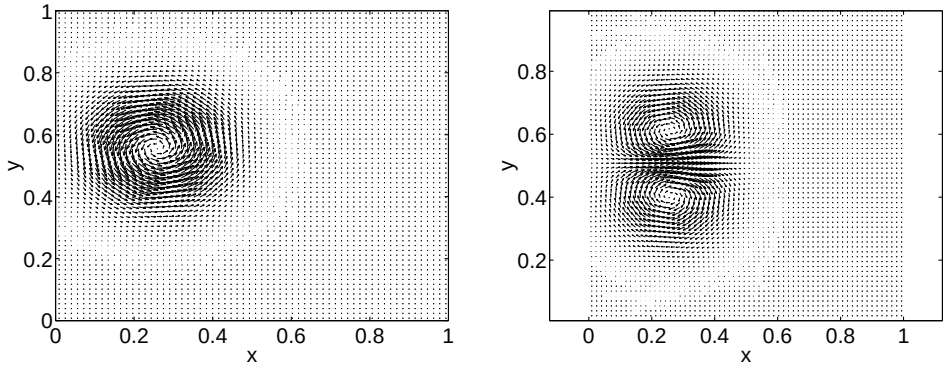


Figure 12.3 Champ de vitesse induit par un seul tourbillon (à gauche) et un dipôle (à droite).

Il faut préciser que le champ de vitesse ainsi construit ne vérifie pas exactement les équations de Navier-Stokes. Il sera corrigé automatiquement par le schéma numérique après le premier pas de temps.

12.7 MISE EN ŒUVRE

La mise en œuvre de ce projet étant assez laborieuse, nous allons construire progressivement les différents modules qui vont constituer le programme final. Chaque module sera validé sur des problèmes tests plus simples.

12.7.1 Résolution d'un système linéaire à matrice tridiagonale et périodique

La forme particulière (*tridiagonale, périodique*) des matrices (12.44, 12.46, 12.56) sera exploitée pour programmer des fonctions efficaces de résolution des systèmes linéaires correspondants. Les programmes seront basés sur l'algorithme de Thomas, présenté d'abord pour les matrices tridiagonales et ensuite pour des matrices tridiagonales et périodiques.

Algorithme 12.5. *Algorithme de Thomas pour la résolution d'un système linéaire à matrice tridiagonale*[5].
Le système :

$$\begin{pmatrix} b_1 & c_1 & 0 & . & . & 0 & 0 \\ a_2 & b_2 & c_2 & 0 & . & 0 & 0 \\ . & . & . & . & . & . & . \\ . & . & . & . & . & . & . \\ 0 & 0 & 0 & 0 & a_{n-1} & b_{n-1} & c_{n-1} \\ 0 & 0 & 0 & 0 & . & a_n & b_n \end{pmatrix} \begin{pmatrix} X_1 \\ X_2 \\ . \\ . \\ X_{n-1} \\ X_n \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ . \\ . \\ f_{n-1} \\ f_n \end{pmatrix}$$

est résolu en introduisant la récurrence :

$$\begin{cases} X_k = \gamma_k - \dfrac{c_k}{\beta_k} X_{k+1}, \quad k = 1 \ldots (n-1) \\ \\ X_n = \gamma_n \end{cases} \tag{12.71}$$

En utilisant ces relations dans le système initial, on peut déterminer les coefficients γ_k *et* β_k *:*

$$\begin{cases} \beta_1 = b_1 \\ \\ \beta_k = b_k - \dfrac{c_{k-1}}{\beta_{k-1}} a_k, \quad k = 2 \ldots n \end{cases}$$

$$\begin{cases} \gamma_1 = \dfrac{f_1}{\beta_1} = \dfrac{f_1}{b_1} \\ \\ \gamma_k = \dfrac{f_k - a_k \gamma_{k-1}}{\beta_k}, \quad k = 2 \ldots n \end{cases}$$

5. On peut démontrer qu'il s'agit d'une forme particulière de l'algorithme de Gauss.

Une fois les coefficients γ_k et β_k calculés, les inconnues X_k seront calculées par substitution inverse en (12.71), à partir de la valeur de X_n.

Algorithme 12.6. *Algorithme de Thomas pour la résolution d'un système linéaire à matrice tridiagonale et périodique.*
Le système

$$\begin{pmatrix} b_1 & c_1 & 0 & . & 0 & 0 & |a_1 \\ a_2 & b_2 & c_2 & . & 0 & 0 & |0 \\ . & . & . & . & . & . & |0 \\ . & . & . & . & . & . & |0 \\ 0 & 0 & 0 & . & a_{n-1} & b_{n-1} & |c_{n-1} \\ -- & -- & -- & -- & -- & -- & -- \\ c_n & 0 & 0 & . & 0 & a_n & |b_n \end{pmatrix} \begin{pmatrix} X_1 \\ X_2 \\ . \\ . \\ X_{n-1} \\ -- \\ X_n \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ . \\ . \\ f_{n-1} \\ -- \\ f_n \end{pmatrix}$$

est écrit sous la forme[6] *:*

$$\begin{pmatrix} b_1^* & c_1 & 0 & . & 0 & 0 & 0 & |v_1 \\ a_2 & b_2 & c_2 & . & 0 & 0 & 0 & |0 \\ . & . & . & . & . & . & . & |0 \\ . & . & . & . & . & . & . & |0 \\ 0 & 0 & 0 & . & a_{n-1} & b_{n-1} & c_{n-1} & |0 \\ 0 & 0 & 0 & . & 0 & a_n & b_n^* & |v_n \\ -- & -- & -- & -- & -- & -- & -- & -- \\ -1 & 0 & 0 & . & 0 & 0 & -1 & |1 \end{pmatrix} \begin{pmatrix} X_1 \\ X_2 \\ . \\ . \\ X_{n-1} \\ X_n \\ -- \\ X^* \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ . \\ . \\ f_{n-1} \\ f_n \\ -- \\ 0 \end{pmatrix} \tag{12.72}$$

avec :

$$\begin{cases} v_1 = a_1 \\ v_n = c_n \\ b_1^* = b_1 - a_1 \\ b_n^* = b_n - c_n \end{cases} \qquad \text{et} \qquad X^* = X_1 + X_n$$

Une forme équivalente de (12.72) est :

$$\underbrace{\begin{pmatrix} b_1^* & c_1 & 0 & . & . & 0 & 0 \\ a_2 & b_2 & c_2 & 0 & . & 0 & 0 \\ . & . & . & . & . & . & . \\ 0 & 0 & 0 & 0 & . & a_n & b_n^* \end{pmatrix}}_{M^*} \begin{pmatrix} X_1 \\ X_2 \\ . \\ X_n \end{pmatrix} + \begin{pmatrix} v_1 \\ 0 \\ . \\ v_n \end{pmatrix} X^* = \begin{pmatrix} f_1 \\ f_2 \\ . \\ f_n \end{pmatrix}$$

et

$$X^* = X_1 + X_n.$$

La solution est cherchée sous la forme :

$$X_k = X_k^{(1)} - X_k^{(2)} \cdot X^*, \quad k = 1 \ldots n \tag{12.73}$$

6. Cette méthode est basée sur une formule similaire à celle de Shermann-Morrison, voir [Joly, 2004].

ce qui nous amène à résoudre deux systèmes à matrice tridiagonale de dimension n :

$$\begin{cases} M^* \cdot X^{(1)} = (f_1\, f_2\, \ldots\, f_{n-1}\, f_n)^T \\ M^* \cdot X^{(2)} = (v_1\, 0\, \ldots\, 0\, v_n)^T \end{cases} \tag{12.74}$$

L'inconnue supplémentaire introduite est calculée par :

$$X^* = \frac{X_1^{(1)} + X_n^{(1)}}{1 + X_1^{(2)} + X_n^{(2)}} \tag{12.75}$$

Pour résumer, les étapes de l'algorithme seront :

- *résoudre les deux systèmes (12.74) par l'algorithme de Thomas (12.5) (cette étape peut être optimisée car les deux systèmes ont la même matrice M^*) ;*
- *calculer X^* de (12.75) ;*
- *trouver la solution par (12.73).*

Exercice 12.1

Écrire une fonction

```
function fi=trid_per_c2D(aa,ab,ac,fi)
```

qui résout simultanément m systèmes à matrice tridiagonale et périodique. L'algorithme 12.6 sera utilisé pour chaque système j (avec $1 \leqslant j \leqslant m$) décrit par les équations :

```
aa(j,i)*X(j,i-1)+ab(j,i)*X(j,i)+ac(j,i)*X(j,i+1)=fi(j,i),
    i=1,...,n
```

avec la condition de périodicité $X(j, 1) = X(j, n)$.
Indications :

– on peut utiliser le calcul vectoriel sous MATLAB pour appliquer simultanément les formules de l'algorithme à tous les m systèmes ; par exemple, le calcul des coefficients b_1^*, b_n^* de la matrice M^* (12.72) s'écrit simplement

```
ab(:,1)=ab(:,1)-aa(:,1);
ab(:,n)=ab(:,n)-ac(:,n);
```

et les calculs seront effectués pour tous les j tels que $1 \leqslant j \leqslant m$.

– les coefficients β_k et γ_k correspondant à la résolution des systèmes (12.74) par l'algorithme de Thomas seront calculés une seule fois, car la matrice est la même pour les deux systèmes.

Tester la fonction en utilisant comme modèle le programme MATLAB *test_trid.m*.[7]

La solution de cet exercice se trouve à la page 271.

12.7.2 Résolution de l'équation instationnaire de la chaleur

Les fonctions nécessaires à l'intégration des équations de Navier-Stokes seront développées dans le cas plus simple de l'équation instationnaire de la chaleur (voir aussi le Projet 1 pour la forme 1D) :

$$\frac{\partial u}{\partial t} - \Delta u(t,x,y) = f(x,y), \quad \text{sur} \quad \Omega = [0,L_x] \times [0,L_y], \tag{12.76}$$

avec des conditions de périodicité dans les deux directions de l'espace et une condition initiale $u(0,x,y) = u^0(x,y)$ donnée. L'équation de la chaleur sera intégrée en temps jusqu'à l'obtention d'un état stationnaire (ou permanent), solution de l'équation :

$$-\Delta u_s(x,y) = f(x,y), \quad \text{sur} \quad [0,L_x] \times [0,L_y], \tag{12.77}$$

avec les mêmes conditions de périodicité. La solution stationnaire $u_s(x,y)$ de l'équation (12.77) est interprétée comme la limite pour $t \to \infty$ de la solution $u(t,x,y)$ de (12.76).

Afin de tester les programmes développés dans ce paragraphe, nous allons considérer

$$f(x,y) = (a^2 + b^2)\sin(ax)\cos(by), \quad \text{avec} \quad a = \frac{2\pi}{L_x},\ b = \frac{2\pi}{L_y}, \tag{12.78}$$

qui vérifie bien les conditions de périodicité. Il est évident que cette forme du second membre a été choisie pour que la solution exacte de l'équation (12.77) soit :

$$u_{ex}(x,y) = \sin(ax)\cos(by). \tag{12.79}$$

La solution numérique obtenue sera systématiquement comparée à cette solution exacte (analytique).

a) Méthode explicite

Pour intégrer en temps l'équation (12.76), le schéma le plus simple est le schéma d'Euler explicite (voir le Projet 1) :

$$u^{n+1} = u^n + \delta t\,(f + \Delta u^n)\,. \tag{12.80}$$

7. Quelques commentaires sur ce programme : les éléments des tableaux `aa,ab,ac` sont définis de manière aléatoire (fonction `rand`) ; pour chaque j, la matrice A du système est reconstituée ; la matrice A est rendue à diagonale dominante pour s'assurer que le système soit inversible ; le second membre du système est calculé par $f = A * \tilde{X}$, avec $\tilde{X}$ imposé ; chaque système est résolu en utilisant les fonctions MATLAB , par la syntaxe $X = A\backslash f$; la fonction `trid_per_c2D` est validée si la solution retournée est exactement $\tilde{X}$.

En supposant que la solution est calculée aux points de coordonnées (x_c, y_m) (figure 12.1), la discrétisation spatiale sera ($i = 1, \ldots, n_{xm}, j = 1, \ldots, n_{ym}$) :

$$u^{n+1}(i,j) = u^n(i,j) + \delta t \left[f(i,j) + \frac{u(ip(i),j) - 2u(i,j) + u(im(i),j)}{\delta x^2} + \frac{u(i,jp(j)) - 2u(i,j) + u(i,jm(j))}{\delta y^2} \right]. \quad (12.81)$$

La solution sera avancée en temps à partir d'une condition initiale $u^0 = 0$, jusqu'à la *convergence* (obtention de la solution stationnaire) exprimée par le critère numérique :

$$\varepsilon = \|u^{n+1} - u^n\|_2 < 10^{-6}, \quad (12.82)$$

avec la norme définie par $\|\varphi\|_2 = \left(\int_\Omega \varphi^2 \, dxdy\right)^{1/2}$.

L'inconvénient de ce schéma est la limitation sur le pas de temps imposée par la condition de stabilité (voir la bibliographie complémentaire) :

$$\delta t \left(\frac{1}{\delta x^2} + \frac{1}{\delta y^2} \right) = \frac{cfl}{2}, \quad cfl \leqslant 1. \quad (12.83)$$

Exercice 12.2

Résoudre numériquement l'équation (12.76) en utilisant le schéma explicite (12.81). Comparer la solution numérique obtenue avec la solution exacte (12.79) en traçant sur une même figure les iso-contours (sous MATLAB , utiliser la fonction `contour`) des deux solutions. Données numériques : $L_x = 1, L_y = 2$, $n_x = 21, n_y = 51, cfl = 1$. Indications :

- les données concernant le maillage peuvent être définies comme variables globales ;
- la programmation doit être modulaire, pour pouvoir utiliser les fonctions par la suite (écrire, par exemple, des fonctions pour calculer f, Δu^n, pour la visualisation, etc.) ;
- utiliser une boucle `while` pour l'avancement en temps ;
- éviter les boucles suivant les indices i et j en utilisant les opérations vectorielles - par exemple la fonction qui calcule Δu^n peut s'écrire (u étant un tableau de dimensions $n_{xm} \times n_{ym}$ et im, jm, ip, jp, les tableaux des indices donnés par 12.30 et 12.29) :

```
function hc=calc_lap(u)
global dx dy
global im ip jp jm ic jc

hc = (u(ip,jc)-2*u+u(im,jc))/(dx*dx)
    +(u(ic,jp)-2*u+u(ic,jm))/(dy*dy);
```

La solution de cet exercice se trouve à la page 271.

Un exemple de résultat est représenté sur la figure 12.4. La méthode explicite est facile à programmer, mais la convergence est lente à cause de la limitation sur le pas de temps et, par conséquent, le temps de calcul est important. Pour remédier à cet inconvénient, nous allons utiliser une méthode implicite pour résoudre le même problème.

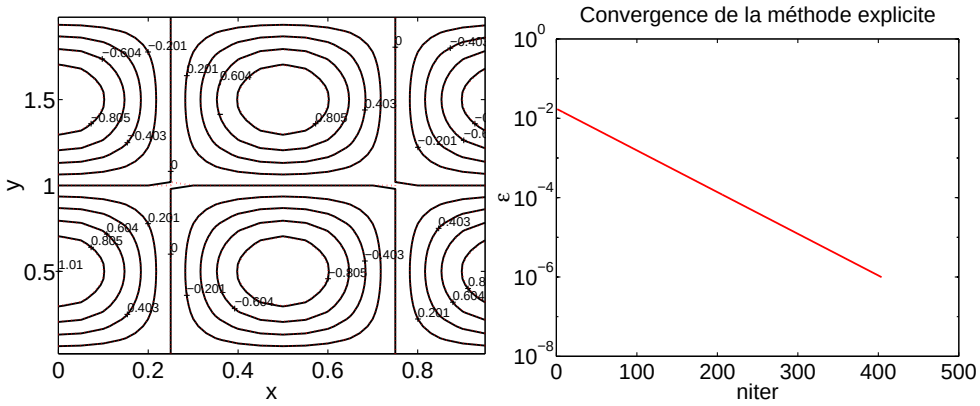


Figure 12.4 Test de la résolution de l'équation stationnaire de la chaleur. Superposition des iso-valeurs de la solution numérique et analytique (à gauche) et convergence de la méthode explicite (à droite).

b) Méthode implicite

Les schémas d'Adams-Bashfort et Crank-Nicolson appliqués à l'équation (12.76) conduisent à :

$$\frac{u^{n+1}-u^n}{\delta t} = \underbrace{\frac{3}{2}\mathcal{H}^n - \frac{1}{2}\mathcal{H}^{n-1}}_{Adams-Bashfort} + \underbrace{\frac{1}{2}\Delta\left(u^{n+1}+u^n\right)}_{Crank-Nicolson} \tag{12.84}$$

avec, dans notre cas $\mathcal{H}^n = \mathcal{H}^{n-1} = f(x,y)$, qui ne dépend pas du temps. L'équation à résoudre devient :

$$\left(I - \frac{\delta t}{2}\Delta\right)\delta u = \delta t\,(f + \Delta u^n)\,, \quad \text{avec} \quad \delta u = u^{n+1} - u^n. \tag{12.85}$$

La factorisation ADI pour cette équation s'écrit sous la forme :

$$\begin{cases} \left(I - \dfrac{\delta t}{2}\dfrac{\partial^2}{\partial x^2}\right)\overline{\delta u} = \delta t\,(f + \Delta u^n) & +\text{périodicité en } x \\ \left(I - \dfrac{\delta t}{2}\dfrac{\partial^2}{\partial y^2}\right)\delta u = \overline{\delta u} & +\text{périodicité en } y. \end{cases} \tag{12.86}$$

La discrétisation de ces équations conduit à des systèmes linéaires à matrice tridiagonale et périodique (voir l'algorithme 12.2) qui seront résolus en utilisant la fonction `trid_per_c2D`, écrite précédemment.

La méthode implicite nécessite plus de calculs pour inverser les systèmes linéaires, mais son coût global est inférieur à celui de la méthode explicite car il n'y a plus de restriction sur le pas de temps δt (la méthode implicite est inconditionnellement stable).

Exercice 12.3

Reprendre l'exercice 12.2 en utilisant la méthode implicite. Le pas de temps sera calculé par la formule (12.83), avec $cfl = 100$! Comparer le temps d'exécution avec celui de la méthode explicite.

En observant que, pour un pas de temps constant, les coefficients des matrices intervenant dans la méthode ADI ne changent pas, optimiser la fonction `trid_per_c2D`

- en réduisant la taille des tableaux qui stockent les coefficients des matrices,
- en effectuant les opérations qui ne dépendent pas du second membre du système une seule fois, en dehors de la boucle (`while`) en temps.

La solution de cet exercice se trouve à la page 271.

Exercice 12.4

Considérons maintenant l'équation de convection-diffusion non-linéaire :

$$\frac{\partial u}{\partial t} + \frac{\partial u^2}{\partial x} - \Delta u = f(x,y), \quad \text{sur} \quad [0, L_x] \times [0, L_y], \tag{12.87}$$

avec des conditions aux limites de périodicité et une condition initiale $u^0(x,y) = 0$.

1. Déterminer l'expression analytique du second membre $f(x,y)$ pour que la solution stationnaire de (12.87) soit (12.79).
2. Retrouver numériquement cette solution en utilisant la méthode implicite (12.84). La seule différence par rapport au programme précédent est le calcul du terme $\mathcal{H}$ (attention, $\mathcal{H}^n \neq \mathcal{H}^{n-1}$).
3. Le pas de temps étant considéré constant et donné par (12.83), trouver la limite de stabilité (cfl_{max}) pour la résolution spatiale donnée.

La solution de cet exercice se trouve à la page 272.

12.7.3 Résolution de l'équation stationnaire de la chaleur en utilisant les FFT

Les modules nécessaires à la résolution de l'équation de Poisson (algorithme 12.4) seront testés dans le cas plus simple de l'équation de la chaleur stationnaire (12.77).

Exercice 12.5

1. Résoudre l'équation (12.77) avec le second membre (12.78) en utilisant la méthode décrite pour la résolution de l'équation de Poisson (page 256) (une transformée de Fourier rapide + la résolution d'un système tridiagonal). Comparer avec la solution exacte.[8]
   Données numériques : $L_x = 1, L_y = 2, n_x = 65, n_y = 129$.
2. Optimiser la partie résolution du système tridiagonal (cf. exercice 12.3).
3. (optionnel) Résoudre la même équation en utilisant deux FFT.

La solution de cet exercice se trouve à la page 272.

12.7.4 Résolution des équations de Navier-Stokes

Les modules les plus importants pour la résolution des équations de Navier-Stokes étant développés, il reste à les assembler pour simuler les écoulements décrits dans le paragraphe 12.6.

Exercice 12.6

Écrire le programme pour la résolution des équations de Navier-Stokes 2D, avec des conditions aux limites de périodicité. Le programme sera modulaire et sera organisé comme suit :

– définition des variables globales ;
– données du programme et définition de la condition initiale ;
– construction du maillage ;
– définition des tableaux pour les variables principales et initialisation à zéro ;
– construction du champ initial en fonction du cas de calcul (voir plus bas) et visualisation ;
– calcul du pas de temps ;
– calcul des variables nécessaires à l'optimisation de la méthode ADI et à l'optimisation de la résolution de l'équation de Poisson ;

8. La solution numérique étant calculée à une constante près, pour comparer avec la solution exacte il faut calculer cette constante en imposant que les deux solutions soient les mêmes en un point choisi ($i = j = 1$, par exemple). Il faut donc comparer u_{ex} et $u_{num} + (u_{ex}(1,1) - u_{num}(1,1))$.

- début boucle en temps :
  - résolution de l'équation de quantité de mouvement pour u ;
  - résolution de l'équation de quantité de mouvement pour v ;
  - calcul de la divergence du champ non-solénoïdal ;
  - résolution de l'équation de Poisson ;
  - correction du champ de vitesse ;
  - calcul de la pression ;
  - calcul du nouveau gradient de pression ;
  - résolution de l'équation pour le scalaire passif ;
  - vérification de la divergence du champ de vitesse ;
  - visualisation de la vorticité et scalaire passif ;
- fin boucle en temps.

La solution de cet exercice se trouve à la page 272.

Cas de calculs[9]

1. Jet plan - instabilité de Kelvin-Helmholtz : paramètres

$$L_z = 2, L_y = 1, n_x = 65, n_y = 65, cfl = 0,2$$

$$Re = 1000, Pe = 1000, U_0 = 1, P_j = 20, R_j = L_y/4, A_x = 0,5, \lambda_x = 0,5L_x.$$

   La condition initiale pour le scalaire passif est identique à celle de la vitesse u.
2. Même configuration que précédemment, sauf $cfl = 0.1, \lambda_x = 0.25L_x$.
3. Dipôle de vorticité :

$$L_z = 1, L_y = 1, n_x = 65, n_y = 65, cfl = 0,4, Re = 1000, Pe = 1000$$

   tourbillon 1 :

$$\psi_0 = +0,01, x_v = L_x/4, y_v = L_y/2 + 0,05, l_v = 0,4\sqrt{2}\min\{x_v, y_v, L_x - x_v, L_y - y_v\}$$

   tourbillon 2 :

$$\psi_0 = -0,01, x_v = L_x/4, y_v = L_y/2 - 0,05, l_v = 0,4\sqrt{2}\min\{x_v, y_v, L_x - x_v, L_y - y_v\}$$

   Le scalaire passif sera initialisé sous la forme d'une bande constante, placée au milieu du domaine de calcul

$$\begin{cases} \chi(i,j) = 1, & \text{si } n_{xm}/2 - 10 \leqslant i \leqslant n_{xm}/2 + 10, \\ \chi(i,j) = 0, & \text{autrement.} \end{cases}$$

4. Même configuration que précédemment + un dipôle similaire qui se propage en sens inverse.
5. Imaginer une autre configuration avec plusieurs dipôles de vorticité.

9. Les résultats qui doivent être obtenus seront illustrés à la fin du projet.

12.8 SOLUTIONS ET PROGRAMMES

Les programmes MATLAB sont organisés en deux répertoires :

- *QP* pour les questions préliminaires (exercices 12.1 à 12.5),
- *QNS* pour la résolution des équations de Navier-Stokes (exercice 12.6).

Un troisième répertoire, *INTERFACE*, contient les programmes réunis, avec une interface graphique qui permet de naviguer facilement entre les différentes questions (un cas de calcul Navier-Stokes supplémentaire est traité également). Pour lancer l'interface graphique, il faut simplement aller dans le sous-répertoire *Tutorial*, lancer le programme *Main* et se laisser guider.

Solution de l'exercice 12.1 : résolution d'un système linéaire à matrice tridiagonale et périodique

La fonction MATLAB *trid_per_c2D.m* résout simultanément m systèmes linéaires à matrice tridiagonale, périodique, de dimension n. Les commentaires permettent de suivre facilement les étapes de l'algorithme 12.6. L'utilisation de la mémoire a été optimisée en utilisant un nombre minimum de tableaux pour les coefficients intermédiaires. Remarquer la programmation vectorielle de l'algorithme. Rappelons que cette fonction est appelée et testée dans le programme *test_trid.m*.

Solution de l'exercice 12.2 : Méthode explicite pour l'équation instationnaire de la chaleur

Le programme *Qexp_lap.m* utilisant la méthode explicite est tout à fait élémentaire et se passe de tout commentaire. Les fonctions utilisées par ce programme sont les suivantes :

- *calc_lap.m* : calcul du laplacien Δu ;
- *fsource.m* : calcul du terme source f ;
- *fexact.m* : calcul de la solution exacte ;
- *norme_L2.m* : calcul de la norme $||u||_2$;
- *visu_isos.m* : visualisation des iso-contours de la solution numérique et de la solution exacte.

Solution de l'exercice 12.3 : Méthode implicite pour l'équation instationnaire de la chaleur

Le programme *Qimp_lap.m* implémente la méthode implicite avec optimisation de la résolution des systèmes à matrice tridiagonale et périodique. Pour cette dernière opération, il suffisait d'observer dans la fonction `trid_per_c2D` (fichier *trid_per_c2D.m*) que tous les calculs qui ne font pas intervenir le second membre `fi` peuvent être faits une seule fois, en dehors de la boucle en temps. C'est le rôle de la fonction `ADI_init` qui retourne les vecteurs `ami, api, alph, xs2` qui

seront ensuite utilisés dans la fonction `ADI_step` pour trouver la solution finale des systèmes linéaires pour chaque second membre `fi` donné. Techniquement, cette optimisation revient à séparer en deux parties la fonction originale `trid_per_c2D`.

Le programme *Qimp_lap.m* garde la même structure que celui écrit pour la méthode explicite, à l'exception de la partie de calcul effectif de la solution.

Solution de l'exercice 12.4 : Méthode implicite pour l'équation de convection-diffusion non-linéaire

Le programme *Qimp_lap_nonl.m* est similaire au programme précédent (*Qimp_lap.m*). La seule différence est le calcul du terme $\mathcal{H}$ à chaque pas de temps. Il s'effectue à l'intérieur de la boucle en temps par l'appel

```
hc =calc_hc(Lx,Ly,xx,yy,u);
```

où la fonction correspondante est écrite dans le fichier *calc_hc.m*. Bien entendu, le terme source f, calculé dans *fsource_nonl.m* a été modifié pour correspondre au problème non-linéaire.

Solution de l'exercice 12.5 : Résolution de l'équation stationnaire de la chaleur en utilisant les FFT

L'algorithme 12.4 est implémenté dans le programme *Qfft_lap.m*. L'étape de résolution du système tridiagonal est optimisée par séparation en deux fonctions `Phi_init` et `Phi_step` ; la différence par rapport à l'optimisation de la méthode ADI est que les coefficients des matrices changent d'un système à l'autre (à cause du nombre d'onde), ce qui nécessite leur stockage dans des matrices.

Solution de l'exercice 12.6 : Résolution des équations de Navier-Stokes

Le programme principal *QNS.m* permet de choisir parmi les cas de calcul suggérés. Les commentaires permettent d'identifier les différentes étapes de l'algorithme de résolution. Les principales fonctions utilisées par ce programme sont celles développées dans les questions préliminaires (`ADI_init`, `ADI_step`, `Phi_init`, `Phi_step`). Les fonctions spécifiques au calcul Navier-Stokes sont les suivantes :

- *init_KH.m* construit la condition initiale pour l'instabilité de Kelvin-Helmholtz (jet plan) ;
- *init_vortex.m*construit la condition initiale pour un tourbillon ; les dipôles de vorticité seront obtenus par la superposition de plusieurs tourbillons ;
- *visu_vort.m* permet de visualiser le champ de vorticité (iso-contours) ;
- *visu_sca.m* visualise les contours du scalaire (traceur) passif ;
- *affiche_div.m* affiche la divergence du champ de vitesse pour vérifier le bon déroulement du calcul (elle doit rester proche de zéro-machine, c'est-à-dire 10^{-15}).

Les figures suivantes illustrent l'évolution des écoulements correspondant aux quatre cas simulés numériquement. Pour les deux premiers cas (figures 12.5 et 12.6), les tourbillons de Kelvin-Helmholtz se forment progressivement, avec une distribution spatiale dictée par la condition initiale (valeur de λ_x/L_x). Si on veut faire un parallèle avec l'écoulement réel, cette simulation numérique (avec des conditions de périodicité) reproduit les phénomènes observables dans une fenêtre de visualisation fixe (la boîte de calcul) qui se déplace en aval avec la vitesse caractéristique de l'écoulement.

Le troisième cas de calcul (figure 12.7) montre l'évolution d'un dipôle de vorticité qui se propage par lui-même suivant l'axe horizontal. Il est intéressant d'observer l'empreinte laissé par le dipôle (grâce à la vitesse induite par la vorticité qu'il contient) dans le scalaire passif qui était initialement au repos. Ce type de structure se rencontre dans plusieurs types d'écoulements fluides, étudiés en océanographie, météorologie ou combustion.

Le dernier cas de calcul (figure 12.8) montre l'interaction frontale entre deux dipôles de vorticité de même intensité. Le résultat est le création de deux nouveaux dipôles (c'est un phénomène de *changement de partenaires*) qui se propagent perpendiculairement à la direction initiale de déplacement. Remarquons que, pour ces deux derniers calculs, les dipôles ne quittent jamais la boîte de calcul, car les conditions de périodicité imposées les font revenir par la frontière opposée (on peut d'ailleurs continuer la simulation pour s'en convaincre).

D'autres cas de calcul peuvent être imaginés suivant le même modèle (prendre, par exemple, une condition initiale avec quatre dipôles de vorticité).

BIBLIOGRAPHIE

[Hirsh, 1988] C. Hirsch : *Numerical computation of internal and external flows*, John Wiley & Sons, 1988.

[Joly, 2004] P. Joly : *Analyse numérique matricielle*, Cassini, Paris, 2004.

[Lesieur, 1994] M. Lesieur : *La Turbulence*, Presses Universitaires de Grenoble, 1994.

[Orlandi, 1999] P. Orlandi : *Fluid Flow Phenomena*, Kluwer Academic Publishers, 1999.

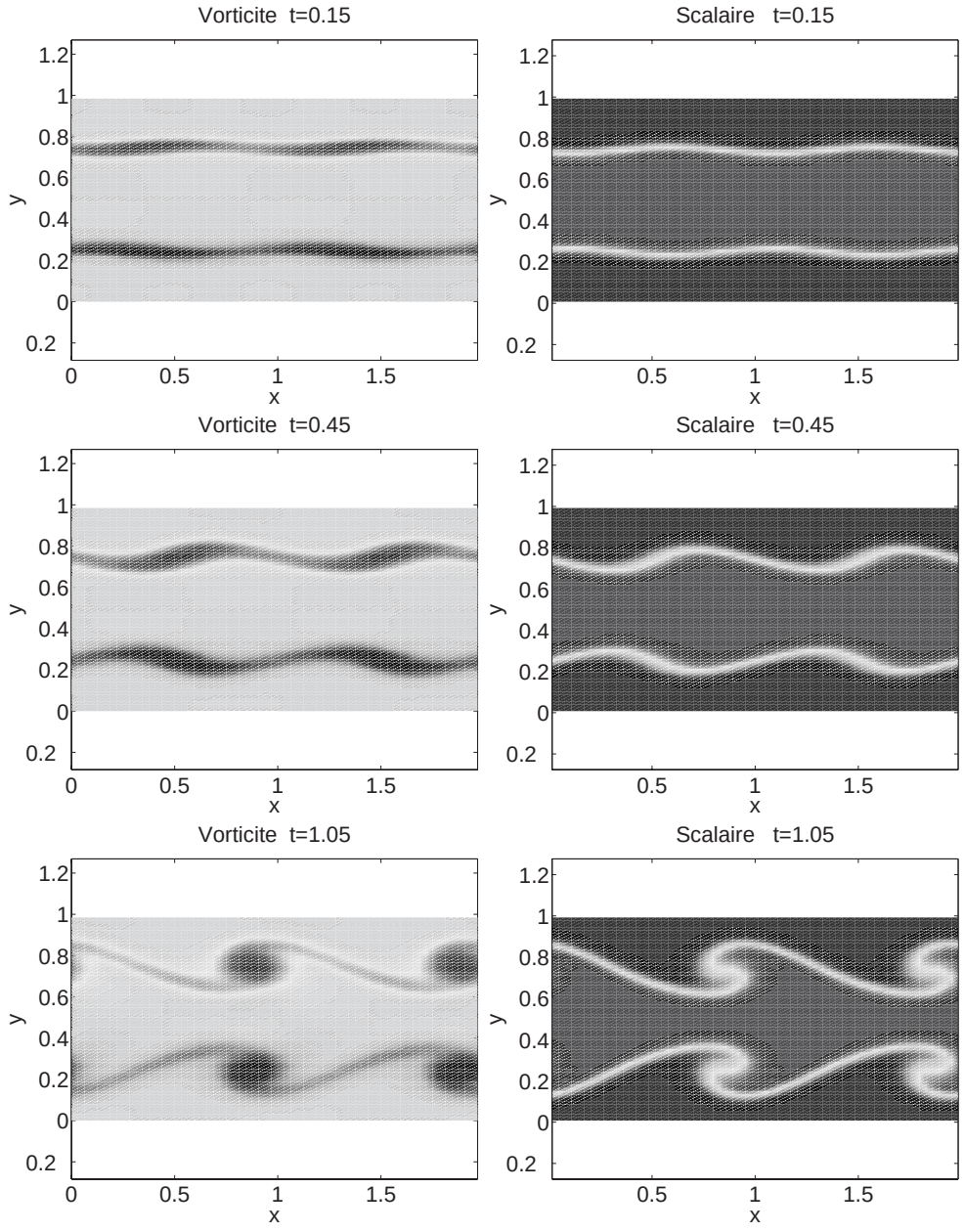


Figure 12.5 Cas de calcul 1. Développement de l'instabilité de Kelvin-Helmholtz pour $\lambda_x / L_x = 0.5$.

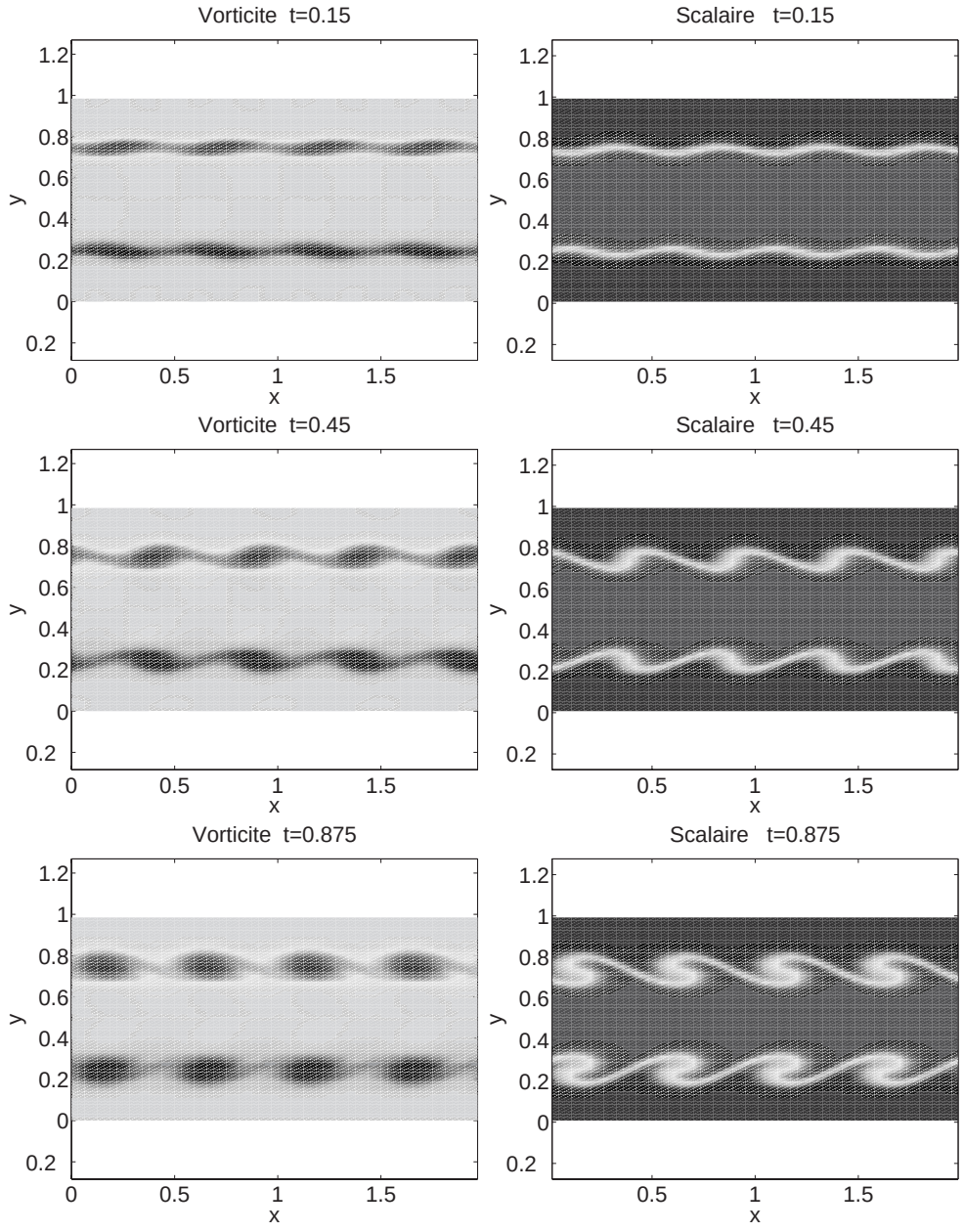


Figure 12.6 Cas de calcul 2. Développement de l'instabilité de Kelvin-Helmholtz pour $\lambda_x/L_x = 0.25$.

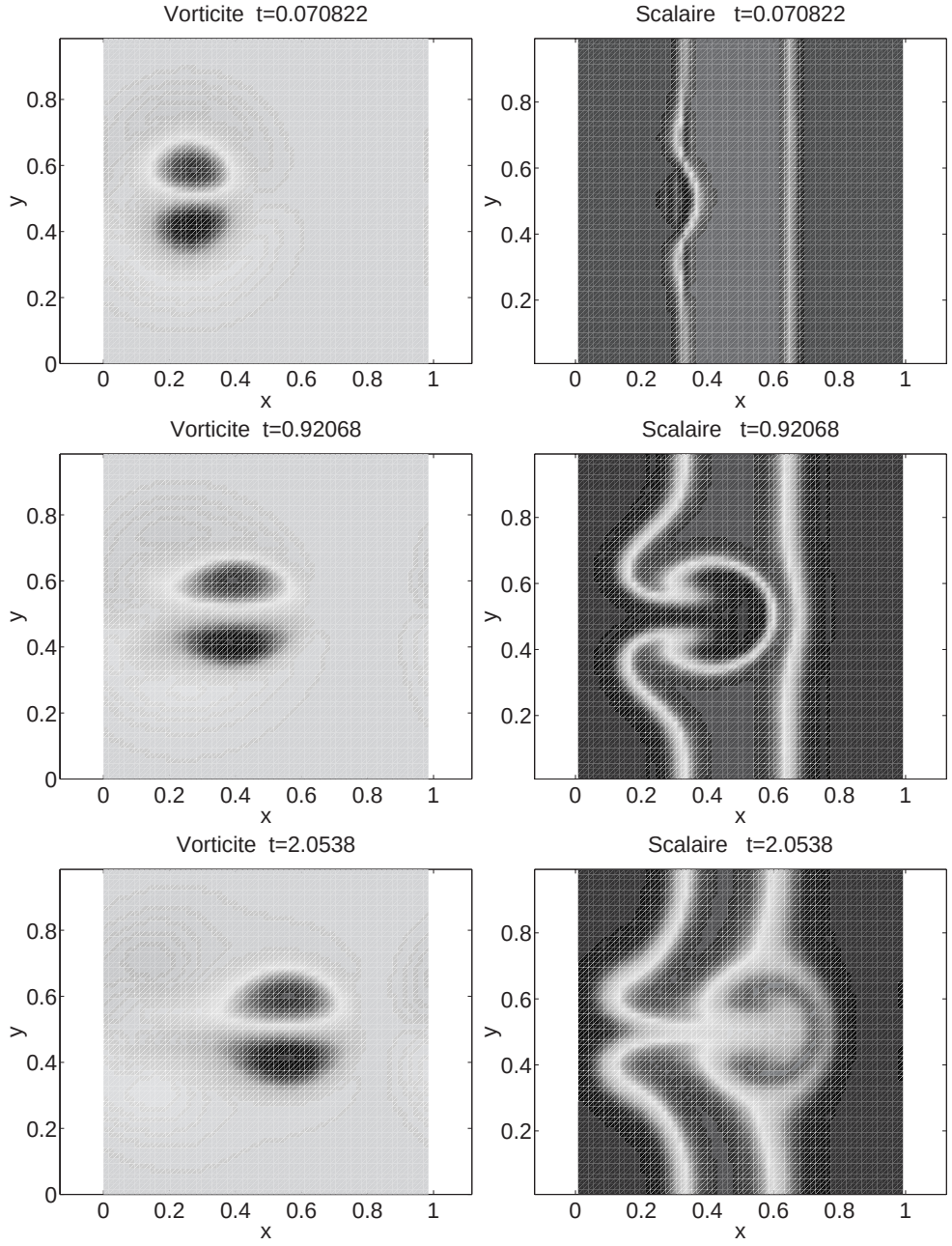


Figure 12.7 Cas de calcul 3. Evolution d'un dipôle de vorticité.

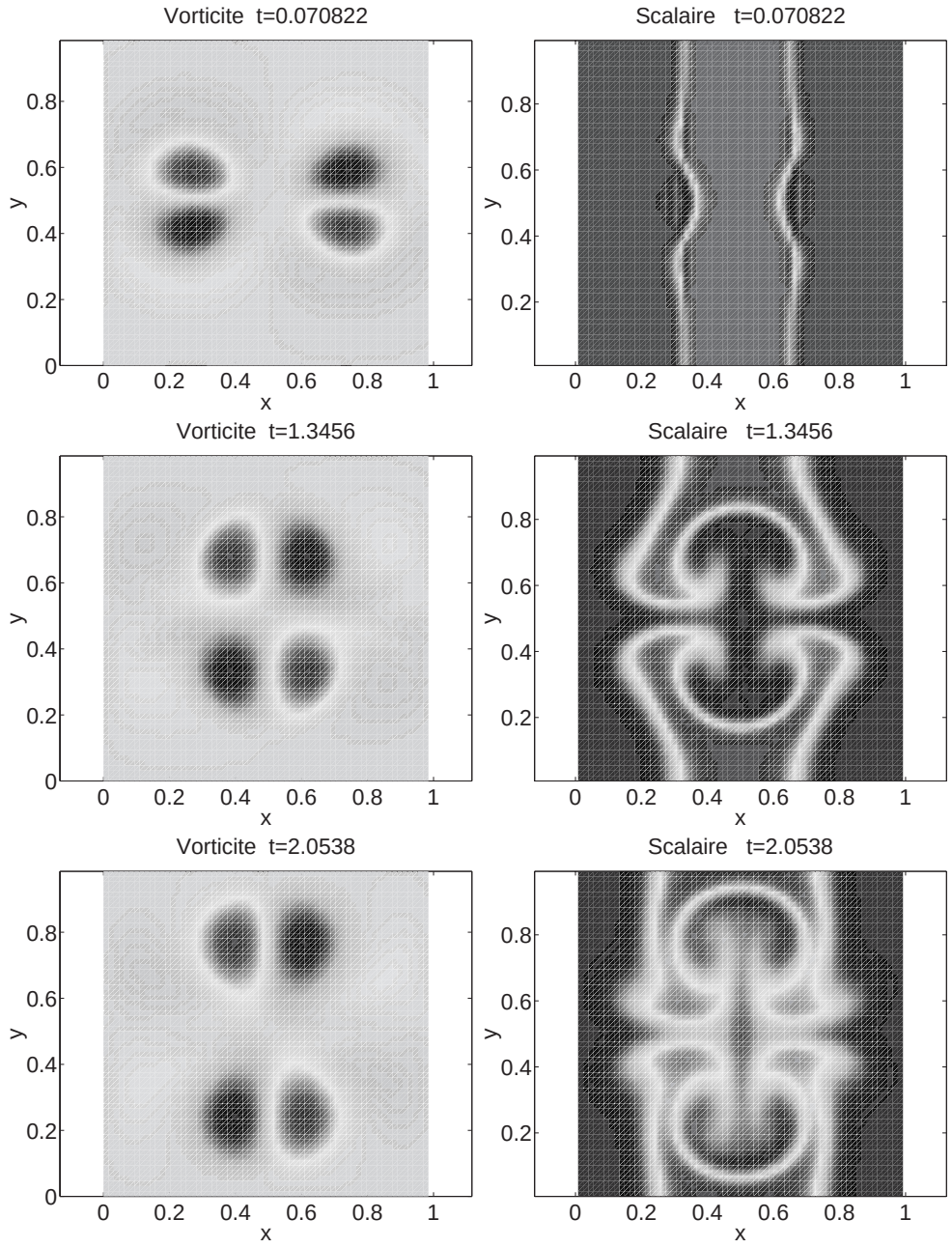


Figure 12.8 Cas de calcul 4. Interaction de deux dipôles de vorticité .

Bibliographie

(**Allaire et Kaber, 2002 (1)**) G. Allaire, S.M. Kaber : *Algèbre linéaire numérique. Cours et exercices*, Ellipses, Paris, 2002.

(**Allaire et Kaber, 2002 (2)**) G. Allaire, S.M. Kaber : *Introduction à Scilab. Exercices pratiques corrigés d'algèbre linéaire*, Ellipses, Paris, 2002.

(**Bernardi, Maday et Rapetti, 2004**) C. Bernardi, Y. Maday, F. Rapetti : *Discrétisations variationnelles de problèmes aux limites elliptiques, collection S.M.A.I. Mathématiques et Applications*, vol. 45, Springer Verlag, Paris, 2004.

(**Bezier, 1986**) P. Bézier : *Courbes et surfaces*, Mathématiques et CAO, vol 4, Hermès, Paris, 1986.

(**Bonnet et Luneau, 1989**) A. Bonnet, J. Luneau : *Aérodynamique : Théories de la Dynamique des Fluides*, Editions Cépaduès, 1989.

(**Brezzi et Russo, 1994**) F. Brezzi, A. Russo : *Choosing bubbles for advection-diffusion problems*, Math. Models and Meth. in Appl. Sci., vol 4, n. 4, 1994.

(**Casteljau, 1985**) P. de Casteljau : *Formes à pôles*, Mathématiques et CAO, vol 2, Hermès, Paris, 1985.

(**Ciarlet, 1982**) P. G. Ciarlet : *Introduction à l'analyse numérique matricielle et à l'optimisation*, Masson, Paris, 1982.

(**Ciarlet, 1978**) P. G. Ciarlet :*The finite element method for elliptic problems*, North Holland. Amsterdam, 1978.

(**Ciarlet, 1986**) P. G. Ciarlet : *Elasticité tridimensionnelle*, Masson, Paris, 1986.

(**Cohen, 2003**) A. Cohen : *Numerical Analysis of Wavelet Methods, Studies in Mathematics and its Applications, 32, North-Holland, Amsterdam, 2003.*

(**Cohen, 1992**) A. Cohen : *Ondelettes et traitement numérique du signal*, Masson, Paris, 1992.

(**Crouzeix et Mignot, 1989**) M. Crouzeix, A. Mignot : *Analyse numérique des équations différentielles*, Masson, Paris, 1989.

(**Danaila, Hecht et Pironneau, 2003**) I. Danaila, F. Hecht, O. Pironneau : *Simulation numérique en C++*, Dunod, Paris, 2003.

(**Daubechies, 1992**) I. Daubechies : *Ten lectures on wavelets*, Society for Industrial and Applied Mathematics. Philadelphia. Pennsylvania, 1992.

(**Delabrière et Postel, 2004**) S. Delabrière, M. Postel : *Méthodes d'approximation. Equations différentielles. Applications Scilab*, Ellipses, Paris, 2004.

(**Demailly, 1996**) J. P. Demailly : *Analyse numérique et équations différentielles*, Presses Universitaires de Grenoble, 1996.

(**Dumas, 1999**) L. Dumas : *Modélisation à l'oral de l'agrégation*, Ellipses, Paris, 1999.

(**Euvrard, 1990**) D. Euvrard : *Différences finies, éléments finis, méthode des singularités*, Masson, Paris, 1990.

(**Farin, 1992**) G. Farin : *Courbes et surfaces pour la CGAO*, Masson, Paris, 1992.

(**Fletcher, 1991**) C. A. J. Fletcher : *Computational Techniques for Fluid Dynamics*, Springer-Verlag, 1991.

(**Godlewski et Raviart, 1996**) E. Godlewski et P.-A. Raviart : *Numerical approximation of hyperbolic systems of conservation laws*, Springer Verlag, 1996.

(**Hairer, Norsett et Wanner, 1987**) E. Hairer, S.P. Norsett, G. Wanner : *Solving Ordinary Differential Equations I, Nonstiff Problems*, Springer series in computational mathematics, 8, Springer Verlag, 1987.

(**Hirsh, 1988**) C. Hirsch : *Numerical computation of internal and external flows*, John Wiley & Sons, 1988.

(**Hoschek et Lasser, 1997**) J. Hoschek, D. Lasser : *Fundamentals of Computer Aided Geometric Design*, Peters. Massachusetts, 1997.

(**Joly, 2004**) P. Joly : *Analyse numérique matricielle*, Cassini, Paris, 2004.

(**Joly, 1990**) P. Joly : *Mise en œuvre de la méthode des éléments finis, collection S.M.A.I. Mathématiques et Applications*, Ellipses, Paris, 1990.

(**Lesieur, 1994**) M. Lesieur : *La Turbulence*, Presses Universitaires de Grenoble, 1994.

(**LeVeque, 1992**) R. LeVeque : *Numerical Methods for Conservation Laws*, Birkhäuser, 1992.

(**Lucquin, 2004**) B. Lucquin : *Équations aux dérivées partielles et leurs approximations*, Ellipses, Paris, 2004.

(**Lucquin et Pironneau, 1996**) B. Lucquin, O. Pironneau : *Introduction au calcul scientifique*, Masson, Paris, 1996.

(**Mallat, 1997**) S.G. Mallat : *A wavelet tour of signal processing*, Academic Press. New York, 1997.

(**Meyer, 1990**) Y. Meyer : *Ondelettes et opérateurs. Tomes I à III*, Hermann, Paris, 1990.

(**Mohammadi et Saïac, 2003**) B. Mohammadi, J.-H. Saïac : *Pratique de la simulation numérique*, Dunod, 2003.

(**Orlandi, 1999**) P. Orlandi : *Fluid Flow Phenomena*, Kluwer Academic Publishers, 1999.

(**Piegl et Tiller, 1995**) L. Piegl, W. Tiller : *The NURBS book*, Springer. Berlin, 1995.

(**Quarteroni et Valli, 1999**) A. Quarteroni, A. Valli : *Domain decomposition methods for partial differential equations*, Numerical Mathematics and Scientific Computation. The Clarendon Press Oxford University Press, New York, 1999. Oxford Science Publications.

(**Risler, 1991**) J.-J. Risler : Méthodes mathématiques pour la CAO, Masson, Paris, 1991.

(**Saad, 1998**) M. Saad : *Compressible Fluid Flow*, Pearson Education, 1998.

(**Schwartz, 1980**) L. Schwartz : *Analyse, topologie générale et analyse fonctionnelle*, Hermann, Paris, 1980.

(**Théodor et Lascaux, 1985**) R. Théodor, P. Lascaux : *Analyse numérique matricielle appliquée à l'art de l'ingénieur*, Masson, Paris, 1985.

Index général

D

E

F

T

U

V

Index des procédures

S

T

V

X

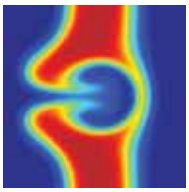
48709 - (I) - (1,2) - OSB 80° - PUB - JME

Achevé d'imprimer sur les presses de
SNEL Grafics sa
rue Saint-Vincent 12 – B-4020 Liège
Tél +32(0)4 344 65 60 - Fax +32(0)4 341 48 41
janvier 2005 — 33566

Dépôt légal : février 2005

Imprimé en Belgique

SCIENCES SUP



Ionut Danaila • Pascal Joly
Sidi Mahmoud Kaber • Marie Postel

INTRODUCTION AU CALCUL SCIENTIFIQUE PAR LA PRATIQUE

12 projets résolus avec MATLAB

Comme leur nom le suggère, les mathématiques appliquées ne peuvent seulement s'enseigner de façon théorique. L'expérimentation numérique est en effet indispensable pour percevoir la puissance, mais aussi les limites, des outils et des méthodes de calcul.

C'est pourquoi cet ouvrage propose douze « projets », basés sur autant de problèmes concrets classiques, qui permettent, grâce à des exercices intermédiaires et des rappels théoriques, de passer de façon progressive des équations aux résultats. Aboutissement de cette démarche pédagogique et pratique, l'ouvrage propose une résolution complète des projets avec MATLAB (les programmes sont intégralement disponibles sur le site web de l'éditeur).

Ce livre doit permettre à tous ceux qui sont confrontés au calcul scientifique – étudiants des écoles d'ingénieur ou de 2e cycle/Master, mais aussi enseignants, chercheurs ou ingénieurs – de comprendre les concepts, les méthodes et les enjeux fondamentaux de la discipline.

Thèmes abordés : Équations aux dérivées partielles et différentielles non linéaires • Schémas numériques (Euler, Runge-Kutta) • Approximation polynomiale • Éléments finis • Différences finies • Méthode spectrale • Analyse multiéchelle, ondelettes • Méthode de Schwarz • Courbes et surfaces de Bézier.

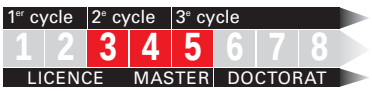
Domaines d'application : Élasticité • Thermique • Mécanique des fluides (équations de Navier-Stokes) • Dynamique des gaz • CAO • Traitement de l'image • Chimie.

Ionut Danaila, Sidi Mahmoud Kaber et Marie Postel sont maîtres de conférences à l'Université Paris 6 Pierre-et-Marie-Curie.

Pascal Joly est ingénieur de recherche au CNRS.

Tous les auteurs appartiennent au laboratoire Jacques-Louis Lions de l'UPMC et au CNRS.

MATHÉMATIQUES
PHYSIQUE
CHIMIE
SCIENCES DE L'INGÉNIEUR
INFORMATIQUE
SCIENCES DE LA VIE
SCIENCES DE LA TERRE



ISBN 2 10 048709 4

www.dunod.com

DUNOD