Titre: Title:	Algorithmes optimaux pour l'approximation de fonctions concaves
Auteur: Author:	Jean Guérin
Date:	2005
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Guérin, J. (2005). Algorithmes optimaux pour l'approximation de fonctions concaves [Thèse de doctorat, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/7558/

UNIVERSITÉ DE MONTRÉAL

ALGORITHMES OPTIMAUX POUR L'APPROXIMATION DE FONCTIONS CONCAVES

JEAN GUÉRIN
DÉPARTEMENT DE MATHÉMATIQUES
ET DE GÉNIE INDUSTRIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

THÈSE PRÉSENTÉE EN VUE DE L'OBTENTION
DU DIPLÔME DE PHILOSOPHIÆ DOCTOR
(MATHÉMATIQUES DE L'INGÉNIEUR)
AOÛT 2005

Your file Votre référence
ISBN: 978-0-494-16995-7
Our file Notre référence
ISBN: 978-0-494-16995-7

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Cette thèse intitulée :

ALGORITHMES OPTIMAUX POUR L'APPROXIMATION DE FONCTIONS CONCAVES

présentée par : GUÉRIN Jean

en vue de l'obtention du diplôme de: Philosophiæ Doctor

a été dûment acceptée par le jury d'examen constitué de :

M. AUDET Charles, Ph.D., président

M. SAVARD Gilles, Ph.D., membre et directeur de recherche

M. MARCOTTE Patrice, Ph.D., membre et codirecteur de recherche

M. DUFOUR Steven, Ph.D., membre

M. DUBEAU François, Ph.D., membre externe

REMERCIEMENTS

Je tiens d'abord à remercier mes deux codirecteurs Gilles Savard et Patrice Marcotte pour leurs conseils et leur soutien.

Je remercie aussi tous les collègues avec qui j'ai eu des discussions à propos de mon travail. Leur intérêt et leurs idées on été grandement appréciés.

Je remercie enfin Nam Soo pour sa patience et ses encouragements tout au long de ce travail.

RÉSUMÉ

Ce travail porte sur l'approximation de fonctions concaves définies sur l'intervalle $[0, 1]$. Celles-ci proviennent d'un sous-ensemble F connu de toutes les fonctions continues sur $[0, 1]$ et ce sous-ensemble constitue de l'information a priori sur les fonctions à approximer. Ici, l'information a priori est la valeur de la fonction et de sa dérivée aux extrémités de l'intervalle. De l'information supplémentaire est disponible sur chaque élément de F, obtenue en évaluant la fonction et l'un de ses surgradients en n points de $[0, 1]$. Ces valeurs sont ensuite utilisées pour construire une approximation de la fonction. La qualité de cette approximation est mesurée par la différence entre celle-ci et la fonction elle-même, évaluée selon l'une des normes L^p avec $1 \leq p \leq \infty$ ou L^1 pondérée par une fonction de poids positive.

L'information obtenue par l'évaluation d'une fonction de F est partielle et ne détermine pas complétement cette fonction. Par conséquent on cherche à choisir les points d'évaluation de façon à minimiser l'incertitude quant à la fonction à approximer, et ce quelque soit l'élément de F qui nous est donné. On est donc amené à déterminer l'information, c'est-à-dire les points d'évaluation, qui minimise l'erreur d'approximation dans le pire cas. Une fois ces points déterminés et la fonction évaluée, on construit ensuite une approximation qui utilise au mieux l'information obtenue. Une procédure pour calculer les points d'évaluation optimaux et construire une approximation à partir de l'information donnée par ceux-ci est appelée *algorithme optimal* pour le problème si elle donne la plus petite erreur d'approximation dans le pire cas.

Les points d'évaluation peuvent être choisis à l'avance quelque soit la fonction, ou encore séquentiellement en utilisant à chaque étape l'information des évaluations précédentes pour choisir le prochain point. Dans le premier cas l'algorithme d'approximation est appelé *passif* et dans le second *adaptatif*. Une question qui se pose alors est de savoir si un algorithme adaptatif est plus efficace qu'un algorithme passif.

La réponse à cette question dépend de tous les détails du problème et est importante en pratique puisque les algorithmes adaptatifs sont habituellement plus coûteux que ceux passifs.

Dans ce document, nous considérons des fonctions concaves sur $[0, 1]$ dont la valeur aux extrémités est fixée, $f(0) = 0$ et $f(1) = 1$, et dont les dérivées en 0 et 1 sont bornées par des constantes connues : $f'(0) \leq a$ et $f'(1) \geq b$. Nous montrons que quelques soient les points d'évaluation choisis la meilleure approximation à l'aide de ces points est une certaine fonction linéaire par morceaux construite à partir des valeurs obtenues. Ainsi, le problème de trouver le meilleur algorithme se réduit à celui de trouver les meilleurs points d'approximation.

Les points d'évaluation optimaux de même que la réponse à la question sur la supériorité des algorithmes adaptatifs dépendent de la norme employée dans la définition de l'erreur d'approximation. Nous montrons que pour l'approximation dans la norme L^1 un algorithme adaptatif ne peut faire mieux dans le pire cas qu'un algorithme passif optimal et un tel algorithme est présenté. Pour l'approximation dans la norme L^p avec $p > 1$ nous montrons qu'un algorithme optimal doit être adaptatif en calculant des bornes explicites sur l'erreur pour les algorithmes passifs et pour un algorithme adaptatif particulier. Nous montrons également que pour l'approximation dans la norme L^1 pondérée, comme dans le cas non pondéré, un algorithme adaptatif ne peut faire mieux qu'un algorithme passif optimal.

Pour chacun de ces problèmes nous proposons des algorithmes optimaux au sens où étant donnés n points d'évaluation aucun autre algorithme ne peut faire mieux, à une constante (indépendante de n) près. Cette emphase sur la minimisation de l'erreur en fonction du nombre d'évaluations est typique de notre approche et reflète le fait qu'en pratique les fonctions à approximer sont souvent très coûteuses à calculer et donc que le nombre de points d'évaluation utilisés est un élément crucial du problème. Nous proposons également pour les problèmes avec les normes L^1 et L^1

pondérée un algorithme séquentiellement optimal, qui est un algorithme adaptatif conçu pour faire le meilleur usage possible de l'information *favorable* obtenue au cours des évaluations.

Enfin, tous les algorithmes que nous construisons sont testés numériquement pour juger de leur mérites respectifs.

ABSTRACT

In this work we consider the approximation of concave functions defined on the interval $[0, 1]$. These come from a known subset F of all continuous functions on $[0, 1]$ and this subset defines *a priori* information about the functions to be approximated. Here, the *a priori* information is the value of the function and its derivative at the endpoints of the interval. Additionnal information is available on each element of F, obtained by evaluating the function and one of its supergradient at n points of $[0, 1]$. These values are then used to construct an approximation of the function. The quality of this approximation is measured by the difference between it and the function itself, under one the L^p norms with $1 \leq p \leq \infty$ or the L^1 norm weighted with a positive function.

The information obtained from the evaluation of a function in F is partial and does not completely determine this function. Therefore, we want to choose the evaluation points so as to minimize the uncertainty on the function being approximated, whichever element of F is actually under consideration. We are thus interested in finding the information, i.e. the evaluation points, that minimize the approximation error in the worst case. Once these points are found, we construct an approximation that makes the best use of the information that was obtained. A procedure to compute the optimal points and construct the best approximation based on these points is called an *optimal algorithm* for the problem if it achieves the smallest error in the worst case.

The evaluation points can be chosen in advance, regardless of the actual function or sequentially, making use at each step of the previous information to choose the next point. In the first case the algorithm is called *passive*, and *adaptive* in the second. A question that arises then is whether an adaptive algorithm is more efficient than a passive one. The answer to that question depends on all the elements of the problem

and it is important in practice because adaptive algorithms are often more expensive than passive ones.

In this document we consider concave functions on $[0,1]$ whose values at the endpoints are fixed : $f(0) = 0$ and $f(1) = 1$ and whose derivatives at 0 and 1 are bounded by known constants : $f'(0) \leq a$ and $f'(1) \geq b$. We show that for any fixed evaluation points the best approximation is a piecewise linear function constructed from the values obtained from these points. Therefore, the problem of finding the best algorithm reduces to that of finding the best evaluation points.

The optimal evaluation points, as well as the answer to the question on the superiority of adaptive algorithms depend on the norm in which the approximation error is measured. We show that for L^1 approximation an adaptive algorithm cannot do better in the worst case than an optimal passive one, and such an algorithm is given. For L^p approximation, with $p > 1$ we show that adaptive algorithms are better by computing error bounds on passive algorithms and for a specific adaptive algorithm. We also show that for weighted approximation, as for L^1 approximation, adaptive algorithms are not better that passive ones.

For each of these problems we construct algorithms which are optimal in the sense that given n evaluation points no other algorithm can do better, up to a constant factor (which does not depend on n). This emphasis on minimizing the error as a function of the number of points is typical of our approach. It reflects the fact that in practice functions are often very costly to compute so therefore the number of evaluations is a crucial factor in the problem. We also construct a sequentially optimal algorithm for the L^1 and weighted L^1 problems. This is an adaptive algorithm designed to make the best use of *favorable* information in the course of the evaluation process.

Finally, all the algorithms we construct are numerically tested and compared.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

INTRODUCTION

Considérons le problème suivant : on cherche à approximer une fonction f d'une seule variable pour laquelle on ne connaît pas de définition analytique mais que l'on peut évaluer en tout point d'un intervalle. La fonction f étant coûteuse à calculer, on veut une approximation construite à partir d'un petit nombre de points d'évaluation. La fonction particulière que l'on doit approximer n'est pas connue mais on suppose que l'on dispose quand même d'information a priori sur f, qui provient du contexte dans lequel se pose le problème. Cette information a priori ne caractérise pas complètement f mais détermine toute une classe de fonctions, qui est l'ensemble de toutes les fonctions partageant la même information que f. Par exemple, dans ce travail la fonction à approximer est concave et sa valeur de même que sa dérivée aux extrémités d'un intervalle fini sont connues. Une définition précise de la classe ainsi déterminée est donnée à la section 1.1.

Notre problème d'approximation peut donc s'énoncer comme suit : connaissant la classe à laquelle appartient la fonction à approximer f, on cherche les points d'évaluation qui donnent le plus d'information sur f de façon à pouvoir reconstruire cette fonction le plus précisément possible. D'un autre point de vue, on peut voir cette classe de fonctions comme un espace d'où on tire des échantillons et on cherche les points d'échantillonnage qui permettent de déterminer f avec la plus petite incertitude possible.

Pour bien définir ce problème il faut, en plus d'information a priori, une mesure de l'erreur d'approximation (de l'incertitude) que l'on veut minimiser. Ainsi, la notion de «meilleurs points d'évaluation» prend un sens précis. Cet aspect du problème est discuté en détails à la section 1.3.

Motivations

La motivation originale pour notre étude de ce type d'approximation provient du problème d'équilibre bicritère sur un réseau. La résolution numérique de ce problème proposée par Marcotte, Nguyen et Tanguay [22] requiert la solution répétée d'un problème de plus court chemin paramétrique. Ceci revient à déterminer la fonction valeur définie par

$$f(\alpha) = \min_i \{C_i(\alpha)\}$$

où C_i est le coût du i^e chemin en fonction du paramètre $\alpha \in \mathbf{R}$. Pour le problème d'équilibre bicritère le coût d'un chemin est de la forme $F_i + \alpha G_i$ où F_i est le temps de parcours, G_i un coût fixe pour l'utilisation du chemin et α donne le poids relatif de ces deux critères et représente la «valeur du temps» pour les usagers. La fonction f est concave et linéaire par morceaux et peut être déterminée complètement en temps fini (voir Chvátal [8]). Cependant pour accélérer l'algorithme proposé dans [22] il a été suggéré de remplacer la valeur exacte de f par une approximation construite à partir d'un petit nombre d'évaluations de la fonction. Chaque évaluation détermine à la fois le plus court chemin pour la valeur de α choisie, la valeur de f en ce point et sa dérivée G_i. Pour être cohérent avec le critère d'arrêt utilisé dans [22], la qualité de l'approximation est mesurée dans la norme L^1. On se retrouve donc dans la situation décrite plus haut.

Cette application est l'une parmi de nombreuses autres. Sonnevend [34] mentionne également des applications de ce type d'approximation en géologie, microscopie, traitement d'images, codage, design et en ingénierie. En mathématiques, il mentionne des liens avec la programmation mathématique, le contrôle optimal et les jeux différentiels. Nous verrons aussi des liens avec la programmation dynamique.

Description des problèmes

Nous donnons ici une brève description informelle des problèmes qui sont étudiés dans notre travail.

Chaque fonction f à approximer est concave et pour simplifier est définie sur [0,1] avec $f(0) = 0$ et $f(1) = 1$. La seule information a priori connue sur f est $f'(0) \leq a$ et $f'(1) \geq b$. Ici $f'(x)$ dénote la dérivée de f en x ou plus généralement un surgradient lorsque la fonction n'est pas dérivable.

Pour approximer f on doit recueillir de l'information supplémentaire. L'information disponible est la valeur de f et de f', qu'il est possible de calculer en tout point $x \in [0, 1]$ et on suppose que n évaluations sont à faire.

La qualité de l'approximation est déterminée par la différence entre celle-ci et f, mesurée dans l'une des normes L^p avec $1 \leq p \leq \infty$ ou la norme L^1 pondérée par une fonction de poids $h > 0$, que nous dénotons par L^1_h. Dans le premier cas, lorsque $p = 1$, le problème d'approximation est équivalent au problème de quadrature, c'est-à-dire l'approximation de l'intégrale de la fonction. Lorsque $p = \infty$, on parle souvent du problème de reconstruction optimale (optimal recovery) de la fonction. Pour l'approximation pondérée, la fonction de poids correspond dans l'exemple de l'équilibre bicritère à la distribution du paramètre α dans la population des usagers du réseau : la valeur du temps n'est pas la même pour tous.

Parmi toutes les fonctions concaves ayant la même information a priori, certaines sont «faciles» à approximer, c'est-à-dire que l'on peut choisir les n points d'évaluation de façon à ce que l'erreur d'approximation soit petite, tandis que d'autres sont plus difficiles. Nous avons choisi de chercher à minimiser l'erreur pour la plus difficile des fonctions de sorte que l'on ait une borne sur l'erreur qui soit valable pour toutes les fonctions de la classe considérée. Notre contexte est donc celui de la minimisation

de l'erreur dans le pire cas.

Les questions auxquelles nous allons répondre sont les suivantes :

1. Pour des points d'évaluation fixés $x_1, \ldots, x_n$ et les valeurs $f(x_1), \ldots, f(x_n)$, $f'(x_1), \ldots, f'(x_n)$ correspondantes, quelle est la meilleure approximation possible de f ?
2. Quels sont les points d'évaluation x_i qui permettent de minimiser l'erreur d'approximation dans le pire cas ?
3. Pour déterminer les points d'évaluation optimaux est-il utile d'utiliser un algorithme adaptatif, qui tient compte des évaluations précédentes pour choisir le point suivant ?

Notre approche dans ce travail est concrète et calculatoire. Notre but est d'énoncer nos problèmes dans un cadre théorique approprié, pour ensuite les exprimer le plus explicitement possible de façon à pouvoir utiliser les résultats théoriques pour construire des procédures de calculs. Chaque algorithme discuté dans notre travail a été implanté et testé numériquement. Les preuves des résultats théoriques sont souvent longues mais ne requièrent que des techniques élémentaires et des calculs détaillés, qui pour la plus grande part ont été effectués à l'aide du logiciel Maple [21].

Organisation du document

Nous décrivons maintenant l'organisation de ce document. Au chapitre 1 nous introduisons les concepts et la notation nécessaires à notre étude. Nous faisons ensuite au chapitre 2 une revue de la littérature sur notre sujet et certains problèmes connexes. Au chapitre 3 nous examinons le problème d'approximation dans la norme L^1. Ce problème peut être complètement résolu et ce de plusieurs façons. Nous en donnons la solution et proposons un algorithme pour le choix des points d'évaluation optimaux. Au chapitre 4, l'approximation dans la norme L^p est abordée. Nous montrons

que pour ce problème les algorithmes adaptatifs sont supérieurs si et seulement si $p > 1$ et nous construisons un algorithme adaptatif d'ordre optimal. Nous discutons également plus en détails dans ce chapitre du cas $p = \infty$, qui est important mais difficile. Au chapitre 5 nous étudions l'approximation L^1 pondérée et nous donnons une condition nécessaire que doivent satisfaire les points d'évaluation optimaux pour ce problème. Cette condition est utilisée au chapitre 6 pour construire un algorithme optimal pour calculer ces points. Dans ce même chapitre sont décrits en détails des algorithmes pour les problèmes L^1, L^∞ et L^1_h. Ces algorithmes sont testés et comparés numériquement au chapitre 7. Finalement nous proposons quelques avenues pour de futures recherches avant de conclure notre travail.

CHAPITRE 1

THÉORIE GÉNÉRALE

Nous développons dans ce chapitre les bases théoriques sur lesquelles est fondé notre travail. Des définitions précises des concepts d'algorithme, erreur d'approximation et optimalité sont données. Ces définitions s'appuient en grande partie sur l'étude systématique des algorithmes optimaux faites par Traub, Wasilkowski et Woźniakowski dans [41, 42, 43] et par Sukharev [40].

1.1 Problèmes considérés

Les problèmes étudiés dans ce travail se posent dans le contexte général suivant. Soit F un sous-ensemble d'un espace linéaire X, B un espace métrique avec une fonction de distance d, et $S : F \to B$ un opérateur. On veut construire un algorithme donnant une approximation $\alpha(f)$ de $S(f)$ pour chaque élément $f \in F$ (voir figure 1.1). De l'information a priori est disponible sur chaque f du fait que l'on sait qu'il s'agit d'un élément de F. De plus, il est possible d'obtenir de l'information supplémentaire pour un élément f donné à l'aide d'un *calcul d'information*.

Deux approches sont possibles pour juger de l'efficacité d'un algorithme. On peut fixer une marge d'erreur $\varepsilon > 0$ et chercher un algorithme qui minimise le nombre de calculs d'information tout en produisant une erreur inférieure à ε sur la classe F. Plus généralement, on cherche à minimiser la complexité informationnelle (information-based complexity, voir Traub *et al.* [43]) pour une marge ε fixée. Une autre façon d'aborder le problème, équivalente à la première, est de fixer le nombre de calculs d'information (ou la complexité) et de chercher un algorithme qui minimise l'erreur

d'approximation. C'est cette deuxième approche qui est adoptée dans ce document.

La notion de complexité est définie de différentes façons par différents auteurs mais essentiellement elle représente le coût du calcul de l'approximation, qui est la somme du coût des calculs d'information et de celui des calculs nécessaires pour la mise en oeuvre de l'algorithme. Dans ce travail nous appellerons *complexité informationnelle* le coût des calculs d'information et *complexité combinatoire* le coût de l'algorithme. Habituellement on suppose que le coût de l'information est beaucoup plus grand que celui de l'algorithme et c'est pourquoi on s'intéresse principalement à la complexité informationnelle. Cependant certains algorithmes optimaux ont une complexité combinatoire très élévée et il faut en tenir compte dans les applications pratiques.

Un problème sera dénoté par un triplet $P = (S, F, B)$ et on parlera du *problème S sur F*. Dans la plupart des situations, F est un ensemble de fonctions et on appellera donc les éléments de F des *fonctions*.

Exemple: Pour les problèmes d'approximation L^p et L_h^1 qui nous intéressent,

$$X = B = L^p[0,1],\ 1 \leq p \leq \infty,$$

$$F = F_{a,b} \equiv \{f : [0,1] \to \mathbf{R} \mid f \text{ concave}, f(0) = 0, f(1) = 1, f'(0) \leq a, f'(1) \geq b, \}$$

$$\text{et} \quad S(f) = f.$$

Ici $f'(0)$ désigne la dérivée à droite de f en $x = 0$ et $f'(1)$ la dérivée à gauche en $x = 1$.

1.2 Algorithme

Soit $P = (S, F, B)$ le problème d'approximer l'opérateur $S : F \to B$, tel que présenté à la section 1.1. Un algorithme $\alpha = (N, \phi)$ est constitué d'un opérateur d'information $N : F \to X \times Y$ et d'une opération terminale $\phi : X \times Y \to B$, et l'approximation de $f \in F$ par α est $\alpha(f) = \phi(N(f))$. L'espace Y est quelconque, mais souvent $Y \subset \mathbf{R}^m$. Quant à X, c'est un espace d'opérateurs sur F. En pratique les éléments de X sont souvent identifiés à des points du domaine de la fonction f à approximer. L'opérateur N représente l'information que l'on peut obtenir sur un élément f et ϕ donne l'approximation de $S(f)$ obtenue à partir de cette information. Ceci est schématisé à la figure 1.1.

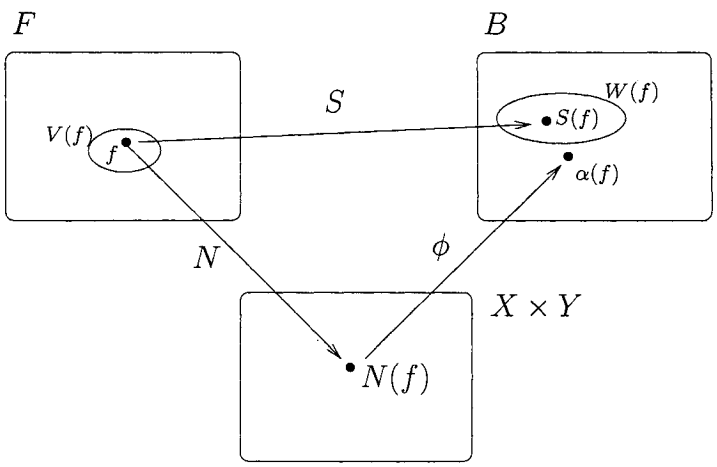


Figure 1.1 Un algorithme général.

Un algorithme comporte deux parties : premièrement le calcul de l'information pour un f donné et deuxièmement la construction de l'approximation de $S(f)$ à l'aide de cette information.

Explicitement, s'il y a n calculs d'information à effectuer, soit $X = X_1 \times \cdots \times X_n$ et $Y = Y_1 \times \cdots \times Y_n$. Alors $N = (\tilde{x}_1, \ldots, \tilde{x}_n)$ où

$$\begin{aligned}
&\tilde{x}_1 \equiv x_1 : F \to Y_1 \\
&\tilde{x}_2 : X_1 \times Y_1 \to X_2 \\
&\vdots \\
&\tilde{x}_n : X_1 \times X_2 \times \cdots X_{n-1} \times Y_1 \times Y_2 \times \cdots \times Y_{n-1} \to X_n
\end{aligned}$$

avec

$$\begin{aligned}
&\tilde{x}_2(x_1, y_1) = x_2, \quad \text{où } x_2 : F \to Y_2 \\
&\vdots \\
&\tilde{x}_n(x_1, \ldots, x_{n-1}, y_1, \ldots, y_{n-1}) = x_n, \quad \text{où } x_n : F \to Y_n.
\end{aligned}$$

N est donc défini comme une suite d'opérateurs $\tilde{x}_i$ qui donnent à chaque étape i une façon de choisir l'information $y_i = x_i(f)$ à recueillir, en fonction de celle déjà obtenue $y_1, \ldots, y_{i-1}$ et des opérateurs $x_1, \ldots, x_{i-1}$ déjà choisis.

Le résultat de la première partie de l'algorithme est l'information $(y_1, \ldots, y_n)$ obtenue successivement comme suit :

$$\begin{aligned}
x_1 &= \tilde{x}_1, & y_1 &= x_1(f) \\
x_2 &= \tilde{x}_2(x_1, y_1), & y_2 &= x_2(f) \\
&\vdots & &\vdots \\
x_n &= \tilde{x}_n(x_1, \ldots, x_{n-1}, y_1, \ldots, y_{n-1}), & y_n &= x_n(f)
\end{aligned}$$

$$\text{et} \quad N(f) = (x_1, \ldots, x_n, y_1, \ldots, y_n).$$

Chaque calcul d'information consiste à choisir d'abord quelle information recueillir, c'est-à-dire x_i, et ensuite à calculer cette information pour la fonction f, c'est-à-

dire y_i.

La deuxième partie de l'algorithme est le calcul de l'approximation de $S(f)$ à partir de $(x_1, \ldots, x_n, y_1, \ldots, y_n)$:

$$\phi(x_1, \ldots, x_n, y_1, \ldots, y_n) = b \in B.$$

On veut ici faire un usage efficace de l'information recueillie pour approximer f. L'opération terminale sera examinée plus en détails à la section 1.5.

La définition donnée ci-dessus est celle d'un algorithme déterministe. On peut également définir le concept d'algorithme stochastique, comme dans Sukharev [40]. Essentiellement un tel algorithme est un choix de mesures de probabilité $\tilde{\sigma}_i$ (à la place des $\tilde{x}_i$) et le choix de x_i est fait aléatoirement selon ces mesures. Nous ne considérons dans ce travail que des algorithmes déterministes, bien qu'en pratique certains éléments aléatoires soient utilisés dans leur implantation. Il s'agit dans ce cas de simplement faire certains choix au hasard lorsqu'un choix optimal n'est pas défini, et ceci ne change pas la nature essentiellement déterministe de l'algorithme.

Pour simplifier l'écriture, nous dénoterons dans la suite par $w^k = (w_1, \ldots, w_k)$ un k-tuplet de nombres w_i.

Exemple: Pour les problèmes d'approximation étudiés ici on a

$$X_i = [0,1], \quad Y_i = \mathbf{R}^2 \qquad \forall i$$

$$\tilde{x}_1 = x_1 \in [0,1], \qquad y_1 = (f(x_1), f'(x_1))$$

$$\tilde{x}_i(x^{i-1}, y^{i-1}) = x_i \in [0,1], \quad y_i = (f(x_i), f'(x_i)) \quad \forall i.$$

Le symbole $f'(x)$ dénote un surgradient de f en x, c'est-à-dire un scalaire ξ tel que $f(z) \leq f(x) + \xi(z - x)$ pour tout $z \in [0, 1]$. Géométriquement, ξ est la pente d'une tangente au graphe de f en x. Un calcul d'information dans ce contexte consiste à choisir un point d'évaluation sur l'intervalle [0,1] puis à calculer la valeur de f et f' en ce point.

Notons ici que $F_{a,b}$ contient des fonctions qui ne sont pas dérivables partout sur [0,1]. Pour ces fonctions, l'opérateur d'information ci-dessus n'est pas bien défini tant qu'un choix du surgradient n'est pas donné. Cette question sera traitée à la section 3.1 où l'on verra que le choix du surgradient n'est pas important, et c'est pourquoi il n'est pas explicité ici.

Les données ci-dessus sont les mêmes pour tous les problèmes que nous considérons. Ce qui distingue ces problèmes est la mesure de l'erreur d'approximation.

1.3 Erreur d'approximation

Étant donné un élément $f \in F$, la mesure de la qualité de l'approximation en f est l'écart entre $S(f)$ et $\alpha(f)$ dans B, c'est-à-dire $d(S(f), \alpha(f))$. On appelle cet écart *l'erreur absolue* de l'algorithme au point f. D'autres mesures sont possibles si B est un espace normé, par exemple l'erreur relative

$$\frac{||S(f) - \alpha(f)||_B}{||S(f)||_B}$$

ou l'erreur normalisée (lorsque F possède une norme)

$$\frac{||S(f) - \alpha(f)||_B}{||f||_F}.$$

Ces deux dernières sont brièvement examinées dans Traub et al. [43]. Dans le présent travail, nous ne considérerons que l'erreur absolue. Pour mesurer la qualité de l'approximation sur toute la classe F, plusieurs approches sont pertinentes. On donne dans Traub et al. [43] les deux suivantes pour les algorithmes déterministes :
PIRE CAS : Ici l'erreur est définie par

$$\sup_{f \in F} d(S(f), \alpha(f)).$$

C'est donc l'erreur absolue pour l'élément f qui est le «pire» pour l'algorithme α.
ERREUR MOYENNE : L'erreur est définie comme étant

$$E\left(d(S(f), \alpha(f))\right),$$

où E est l'espérance par rapport à une mesure de probabilité fixée sur F.

Dans ce travail, seule l'erreur dans le pire cas est considérée. L'erreur moyenne est aussi une mesure pertinente pour notre problème mais requiert une analyse d'un type différent de celle faite ici. En particulier, il est nécessaire de définir une mesure de probabilité sur F pour pouvoir exprimer l'erreur moyenne d'un algorithme.

Exemple: Pour le problème d'approximation L^p, l'écart entre deux fonctions de $F_{a,b}$ est mesuré dans la norme L^p. Soit $\alpha(f)(t)$ la valeur de l'approximation $\alpha(f)$ en t. L'erreur d'approximation est

$$\begin{cases} \| f - \alpha(f) \|_p = \left(\int_0^1 | f(t) - \alpha(f)(t) |^p \, dt \right)^{\frac{1}{p}}, & \text{si } 1 \leq p < \infty \\ \| f - \alpha(f) \|_\infty = \max\limits_{t \in [0,1]} | f(t) - \alpha(f)(t) |, & \text{si } p = \infty. \end{cases}$$

Pour l'approximation L_h^1 avec une fonction de poids $h : [0,1] \to \mathbf{R}^+$ l'erreur est

$$|| f - \alpha(f) ||_{1,h} = \int_0^1 | f(t) - \alpha(f)(t) | h(t)\, dt.$$

Par la suite, lorsque la discussion sera valable pour toutes les normes considérées, nous utiliserons la notation $|| \cdot ||$.

1.4 Information

Une caractéristique essentielle des problèmes considérés ici est que l'information donnée par l'opérateur N est partielle, c'est-à-dire que N n'est pas injectif. On peut donc avoir $f, \tilde{f} \in F$ avec $N(f) = N(\tilde{f})$ et $\phi(N(f)) = \phi(N(\tilde{f}))$, bien que $S(f) \neq S(\tilde{f})$. Pour Traub et al. [43] ceci distingue l'approche de la complexité informationnelle de celle de l'approximation classique. Dans le premier cas, l'approximation est construite à partir d'information partielle tandis que dans le second cas l'information est complète. Il arrive fréquemment en pratique que l'information disponible ne soit que partielle, et c'est le cas pour nos problèmes d'approximation, ce qui justifie l'étude du point de vue de la complexité informationnelle.

L'ensemble des opérateurs d'information que l'on peut utiliser est habituellement un sous-ensemble de l'ensemble I de toutes les applications de F vers $X \times Y$. Cette restriction dépend des caractéristiques du problème, c'est-à-dire de l'information que l'on peut calculer en pratique et représenter par un vecteur de dimension finie. Pour un problème donné, l'ensemble des opérateurs d'information admissibles sera noté $\hat{I}$.

Dans l'étude de Traub et al. le concept de *rayon d'information* joue un rôle crucial.

Il est défini comme suit. Pour f fixé, soit

$$V(f) = \left\{ \tilde{f} \in F \,|\, N(f) = N(\tilde{f}) \right\},$$

l'ensemble de tous les éléments de F ayant la même information que f, et soit

$$W(f) = S(V(f))$$

l'image de cet ensemble dans B. Le rayon de l'ensemble $W(f)$ est défini par

$$\text{ray}(W(f)) = \inf_{a \in B} \sup_{\tilde{f} \in V(f)} d(a, S(\tilde{f})).$$

Informellement, $\text{ray}(W(f))$ est le rayon de la plus petite boule dans B contenant $W(f)$. Ceci est illustré à la figure 1.2. Le rayon d'information de l'opérateur N est le supremum sur tous les $f \in F$ de ces rayons locaux. On définit aussi le diamètre d'information de N qui est le maximum des diamètres des ensembles $W(f)$.

Définition 1.1 *Le rayon d'information de l'opérateur N pour le problème S sur F est*

$$r(N, S) = \sup_{f \in F} \text{ray}(W(f)).$$

Le diamètre d'information de N pour le problème S sur F est

$$D(N, S) = \sup_{\substack{f \in F \\ \tilde{f}, \tilde{g} \in V(f)}} d(S(\tilde{f}), S(\tilde{g})).$$

Ces deux quantités sont liées par la relation

$$r(N, S) \leq D(N, S) \leq 2r(N, S),$$

Figure 1.2 Le rayon r et le diamètre D de $W(f)$.

prouvée dans Traub et al. [41]. Notons que $D(N,S)$ est souvent plus facile à calculer que $r(N,S)$. Par la suite, lorsque S et F sont fixés on notera simplement $r(N)$ et $D(N)$.

Le rayon d'information est une mesure de l'incertitude intrinsèque associée à l'information donnée par N. Ce rayon donne une borne inférieure sur l'erreur (dans le pire cas) d'un algorithme utilisant l'information N.

Théorème 1.1 *[41] Pour toute opération terminale ϕ,*

$$\sup_{f\in F} d(\, S(f), \phi(N(f))\,) \geq r(N,S).$$

Un opérateur d'information est optimal s'il minimise l'erreur dans le pire cas.

Définition 1.2 *Pour ϕ fixé, l'opérateur d'information N^* est optimal si*

$$\sup_{f\in F} d(\, S(f), \phi(N^*(f))\,) = \inf_{N\in\hat{I}} \sup_{f\in F} d(\, S(f), \phi(N(f))\,).$$

Pour le problème d'approximation L^1, le rayon d'information minimal, correspondant à l'information optimale, est donné à la section 3.2. Pour l'approximation L^p avec $p > 1$, une borne supérieure sur le rayon minimal est donnée à la section 4.3.

1.5 Opération terminale

L'opération terminale ϕ est construite à partir de l'information $N(f)$ pour tout $f \in F$. Ici aussi le choix de cette opération est habituellement restreint à un sous-ensemble, dénoté $\hat{T}$, de toutes les opérations terminales possibles. La théorie classique de l'approximation fournit plusieurs résultats sur la façon d'utiliser au mieux l'information disponible pour l'approximation de f étant donné $N(f)$. Un cas particulier est lorsque $\hat{T}$ contient une unique opération permise, ce qui sera le cas pour nos problèmes d'approximation. On pourrait aussi restreindre $\hat{T}$ aux opérations linéaires, etc.

Pour un opérateur d'information fixé, une opération terminale est optimale si elle minimise l'erreur d'approximation.

Définition 1.3 *Pour $N \in \hat{I}$ fixé, $\phi^* \in \hat{T}$ est optimale si*

$$\sup_{f \in F} d(\, \phi^*(N(f)), S(f)\,) = \inf_{\phi \in \hat{T}} \sup_{f \in F} d(\, \phi(N(f)), S(f)\,).$$

Une opération terminale qui est optimale pour un N donné ne l'est pas nécessairement pour un opérateur d'information différent. Cependant une opération terminale centrale, telle que définie ci-dessous, possède la propriété d'être optimale quelle que soit l'information N.

Un *centre* de $W(f) \subset B$ est un élément $c \in B$ tel que

$$\sup_{a \in W(f)} d(a, c) = \inf_{b \in B} \sup_{a \in W(f)} d(a, b).$$

Ceci est illustré à la figure 1.2, où c est un centre de $W(f)$.

Définition 1.4 *Une opération terminale ϕ est centrale si pour tout $N \in \hat{I}$ et $f \in F$ $\phi(N(f)) = c(f)$ est un centre de $W(f) \subset B$.*

Il est démontré dans Sukharev [40] que s'il n'y a aucune restriction sur les opérations terminales admissibles alors il existe une opération qui est centrale. L'intérêt d'une telle opération est qu'elle est optimale pour tout N, et en plus elle est la meilleure opération terminale possible pour chaque $f \in F$.

Théorème 1.2 *[40] Si $\phi_c \in \hat{T}$ est centrale alors*

1. *Pour tout $N \in \hat{I}$*

$$\sup_{f \in F} d(\, \phi_c(N(f)), S(f)\,) = \inf_{\phi \in \hat{T}} \sup_{f \in F} d(\, \phi(N(f)), S(f)\,)$$

2. *Pour tout $N \in \hat{I}$ et $f \in F$*

$$\sup_{\tilde{f} \in V(f)} d(\phi_c(N(f)), S(\tilde{f})) = \inf_{\phi \in \hat{T}} \sup_{\tilde{f} \in V(f)} d(\phi(N(f)), S(\tilde{f}))$$

c'est-à-dire que ϕ_c minimise l'erreur locale en f (voir section 1.6).

Exemple: Pour les problèmes d'approximation L^p et L_h^1 on définit l'opération terminale ϕ^* par

$$\phi^*(N(f)) = \frac{1}{2}(U + L)$$

où

$$U(t) = \sup_{\tilde{f} \in V(f)} \tilde{f}(t), \quad t \in [0,1]$$

$$L(t) = \inf_{\tilde{f} \in V(f)} \tilde{f}(t), \quad t \in [0,1].$$

Notons que U et L dépendent de f, même si cette dépendance n'est pas dénotée

explicitement. Nous verrons à la section 3.1 que cette opération terminale est centrale et donc optimale. Ceci justifie la restriction de $\hat{T}$ à cette unique opération pour nos problèmes. Géométriquement, les fonctions U et L correspondent aux enveloppes supérieure et inférieure de l'ensemble de toutes les fonctions de $F_{a,b}$ ayant la même information que f (voir figure 1.3). Une définition explicite de ces fonctions est donnée à la section 3.2.

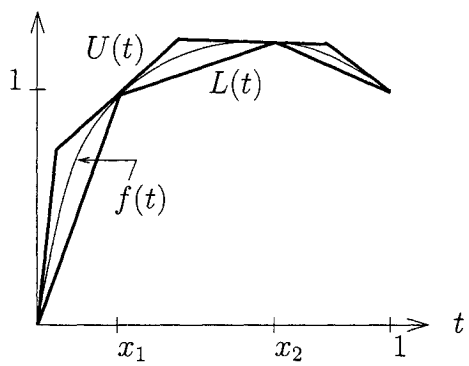


Figure 1.3 Les fonctions L et U.

1.6 Algorithmes optimaux

Le choix d'un algorithme pour résoudre un problème se fait en trois étapes :

1. Formulation du problème, c'est-à-dire une définition précise de F, B et S, ainsi que de la mesure de l'erreur.
2. Choix de l'opérateur d'information. Sur le plan pratique, on doit définir l'ensemble des opérateurs admissibles. Du point de vue théorique on doit, par exemple, déterminer si un algorithme adaptatif (voir section 1.7) est susceptible de donner de meilleurs résultats.
3. Choix de l'opération terminale ϕ.

Plusieurs objectifs sont à considérer lors du choix d'un algorithme. On considère dans Traub et al. [41] les suivants :

- Pour F, B, S, ϕ donnés, trouver l'opérateur d'information optimal.
- Pour F, B, S, N donnés, trouver l'opération terminale optimale.
- Pour F, B, S donnés, trouver un algorithme optimal pour l'erreur (*optimal by error*), c'est-à-dire un algorithme qui réduit au maximum l'erreur d'approximation.
- Pour F, B, S donnés, trouver un algorithme avec une erreur fixée dont la complexité (combinatoire ou informationnelle) est minimale.

Clairement, certains de ces objectifs peuvent être contradictoires. Par exemple, un algorithme optimal pour l'erreur peut avoir une complexité élevée. Dans ce travail nous avons choisi l'objectif suivant, qui correspond au troisième critère décrit ci-dessus :

> Trouver un algorithme α^* qui minimise l'erreur dans le pire cas étant donné que n calculs d'information sont permis.

Ici, *calcul d'information* signifie le calcul d'un point puis l'évaluation de la fonction à approximer et d'un de ses surgradients en ce point. Cet objectif est cohérent avec la situation décrite en introduction, à savoir que le coût d'une évaluation est très élevé par rapport aux calculs servant à déterminer les points d'évaluation et à calculer l'approximation. C'est aussi l'objectif retenu habituellement pour les problèmes d'approximation dans ce contexte.

Avant de définir la notion d'algorithme optimal, nous introduisons la notation suivante, qui sera employée dans la suite. D'abord dénotons par $\hat{A} \subset \hat{I} \times \hat{T}$ l'ensemble des algorithmes admissibles pour un problème donné. Ensuite, soit

$$e(\alpha, f) = d(\alpha(f), S(f))$$

l'erreur de l'algorithme α sur l'élément $f \in F$. Lorsque N ou ϕ est fixé on écrit plutôt $e(\phi, f)$ ou $e(N, f)$.

L'erreur dans le pire cas de α sur F est

$$e(\alpha) = \sup_{f \in F} e(\alpha, f)$$

et *l'erreur locale* de α en f est

$$e_{loc}(\alpha, f) = \sup_{\tilde{f} \in V(f)} d(\alpha(f), S(\tilde{f})).$$

Définition 1.5 *Un algorithme α^* est optimal dans le pire cas s'il satisfait*

$$e(\alpha^*) = \inf_{\alpha \in \hat{A}} e(\alpha).$$

Le résultat suivant sera utile plus tard :

Lemme 1.1 *L'erreur dans le pire cas est donnée par*

$$e(\alpha) = \sup_{f \in F} e_{loc}(\alpha, f).$$

Preuve: D'abord, puisque $f \in V(f)$ pour tout f on a

$$\sup_{\tilde{f} \in V(f)} d(\alpha(f), S(\tilde{f})) \geq d(\alpha(f), S(f))$$

ce qui implique que

$$\sup_{f \in F} e_{loc}(\alpha, f) \geq \sup_{f \in F} d(\alpha(f), S(f)) = e(\alpha).$$

Ensuite, si $\tilde{f} \in V(f)$ alors $\alpha(\tilde{f}) = \alpha(f)$ donc pour tout $f \in F$

$$\begin{aligned} \sup_{\tilde{f} \in V(f)} d(\alpha(f), S(\tilde{f})) &= \sup_{\tilde{f} \in V(f)} d(\alpha(\tilde{f}), S(\tilde{f})) \\ &\leq \sup_{\tilde{f} \in F} d(\alpha(\tilde{f}), S(\tilde{f})) = e(\alpha), \end{aligned}$$

ce qui implique que $\sup_{f \in F} e_{loc}(\alpha, f) \leq e(\alpha)$ et le lemme est démontré. ∎

1.7 Algorithmes passifs et adaptatifs

1.7.1 Information séquentielle et non-séquentielle

Dans la première phase de l'algorithme défini à la section 1.2, si les applications $\tilde{x}_i$ ne dépendent pas de f, c'est-à-dire que $\tilde{x}_i \equiv x_i \in X_i$, l'information peut être calculée simultanément pour tout i. On peut alors considérer que les x_i sont choisis à l'avance et ne dépendent que de l'information a priori $f \in F$. Dans ce cas, l'algorithme α qui en résulte est appelé *passif* ou *non-séquentiel.*

Dans le cas où les $\tilde{x}_i$ dépendent de l'information sur f, les calculs d'information de la première phase doivent être effectués de façon séquentielle : il faut calculer $x_j(f)$ pour $j \leq i$ pour obtenir $\tilde{x}_{i+1}$. L'algorithme est alors *adaptatif* ou *séquentiel.*

Puisqu'un algorithme séquentiel utilise plus d'information et s'adapte à la fonction à approximer, il est possible qu'il soit plus efficace qu'un algorithme passif. Cependant nous le verrons que ça n'est pas toujours le cas. Un algorithme adaptatif est plus difficile à mettre en oeuvre qu'un algorithme passif car il faut alors, en plus de l'évaluation $x_i(f)$, une procédure pour déterminer les fonctionnelles $\tilde{x}_i$ à chaque étape et cette procédure peut être compliquée. De plus, un algorithme passif possède

l'avantage que les calculs d'information peuvent être faits en parallèle, ce qui réduit encore le temps d'exécution. Il est donc pertinent de se demander si pour un problème donné un algorithme adaptatif est plus efficace qu'un algorithme passif.

La réponse à cette question dépend du problème, c'est-à-dire de F, B et S, du type d'algorithme utilisé, déterministe ou stochastique, du contexte, pire cas ou cas moyen, et de la mesure de l'erreur. Un petit changement dans la formulation du problème peut entraîner une réponse différente. Par exemple, pour certains problèmes les algorithmes adaptatifs ne sont pas plus efficaces dans le pire cas mais le sont dans le cas moyen (voir par exemple Novak [25]). De même, les algorithmes adaptatifs peuvent être plus efficaces dans le cas stochastique mais pas dans le cas déterministe (voir Novak [25] et Sukharev [39]).

Sur ce point, l'exemple donné par Sonevend [33] est éclairant : pour la même classe de fonctions convexes sur [0,1] deux sous-ensembles sont définis en fixant la valeur des fonctions en $x = 1/2$ pour la première et $x = 1/3$ pour la deuxième, en plus des deux extrémités. On peut de plus évaluer $f(x)$ en tout point de $[0, 1]$. Un algorithme adaptatif pour l'approximation L^1 est plus efficace pour le second problème mais pas pour le premier.

Pour nos problèmes d'approximation, nous verrons qu'un algorithme adaptatif est plus efficace seulement dans le cas où l'erreur est mesurée dans la norme L^p avec $p > 1$. Pour les cas L^1 et L_h^1 il existe des algorithmes passifs optimaux.

1.7.2 Utilité de l'adaptation

Dénotons par $\hat{A}_{\text{pas}}$ l'ensemble des algorithmes admissibles passifs et par $\hat{A}_{\text{ad}}$ l'ensemble des algorithmes admissibles adaptatifs. On définit

$$e_{\text{pas}} = \inf_{\alpha \in \hat{A}_{\text{pas}}} e(\alpha) \quad \text{et} \quad e_{\text{ad}} = \inf_{\alpha \in \hat{A}_{\text{ad}}} e(\alpha),$$

l'erreur optimale pour les algorithmes passifs et adaptatifs, respectivement. Les quantités e_{pas} et e_{ad} sont des bornes inférieures sur l'erreur maximale pour un problème donné. Puisque les algorithmes passifs sont des cas particuliers d'algorithmes séquentiels, on a $\hat{A}_{\text{pas}} \subset \hat{A}_{\text{ad}}$ et

$$e_{\text{ad}} \leq e_{\text{pas}}.$$

Pour qu'il soit intéressant d'utiliser un algorithme adaptatif pour un certain problème, il faut que e_{ad} soit inférieur à e_{pas} de façon significative.

Plus précisément, on dira que *l'adaptation est utile (adaption helps)* si

$$e_{\text{pas}} = \mathcal{O}(n^{\varepsilon} e_{\text{ad}})$$

pour $\varepsilon > 0$, où n est le nombre de calculs d'information. Si $e_{\text{pas}} > e_{\text{ad}}$ mais $e_{\text{pas}} = \mathcal{O}(e_{\text{ad}})$, on dira que *l'adaptation n'est pas utile au sens large.* Dans ce cas, l'erreur pour les algorithmes passifs et adaptatifs optimaux est du même ordre, mais ces derniers peuvent être légèrement meilleurs. Si $e_{\text{pas}} = e_{\text{ad}}$ alors on dira que *l'adaptation n'est pas utile au sens strict.* Dans ce cas, un algorithme adaptatif ne peut faire mieux dans le pire cas qu'un algorithme passif optimal.

1.7.3 Résultats généraux

L'utilité de l'adaptation pour un problème dépend de tous les paramètres de ce problème. Cependant, il est possible de montrer certains résultats généraux sur l'utilité de l'adaptation.

Théorème 1.3 *[39] Pour un problème donné si e_{loc} possède un point de selle généralisé, c'est-à-dire que*

$$\sup_{f\in F}\inf_{\alpha\in\hat{A}} e_{loc}(\alpha, f) = \inf_{\alpha\in\hat{A}}\sup_{f\in F} e_{loc}(\alpha, f)$$

alors $e_{pas} = e_{ad}$, l'adaptation n'est pas utile au sens strict pour ce problème.

On montre aussi dans Gal et Micchelli [11] les corollaires suivants :

Corollaire 1.1 *[11] S'il existe $f_0 \in F$ tel que*

$$e_{loc}(\alpha, f_0) \geq e_{loc}(\alpha, f), \quad \forall\, \alpha, f$$

alors $e_{pas} = e_{ad}$.

L'élément f_0 doit être vu comme un «pire»élément pour le problème. Si on considère un algorithme séquentiel comme un jeu à deux joueurs, le premier qui cherche à minimiser l'erreur et le deuxième qui veut la maximiser, f_0 est une stratégie optimale pour ce dernier, quelle que soit la stratégie du premier.

Si F est convexe et symétrique par rapport à f_c alors le centre f_c est un pire élément et le corollaire 1.1 s'applique :

Corollaire 1.2 *[11] Si F est convexe et symétrique alors $e_{pas} = e_{ad}$.*

1.8 Optimalité séquentielle

Un algorithme optimal dans le pire cas tel que défini ci-dessus donne, sous la forme d'une borne supérieure, une garantie sur l'erreur d'approximation pour toutes les fonctions de la classe F. Cependant un tel algorithme n'utilise pas nécessairement l'information obtenue sur la fonction à approximer de façon optimale dans le cas où cette information est favorable. Il est facile de trouver des exemples où un algorithme optimal dans le pire cas est très inefficace pour une fonction f représentant un «bon» cas (pour un exemple, voir Sukharev [38]). Sukharev suggère donc dans [40] de raffiner la définition d'optimalité et introduit le concept d'algorithme *séquentiellement optimal.* Dans la terminologie de l'auteur, un calcul d'information constitue une *étape* de l'agorithme. Un algorithme séquentiellement optimal à n étapes est optimal sur F comme ci-dessus et de plus, pour chaque étape i, est un algorithme optimal à $n-i$ étapes sur la classe $F_i \subset F$ définie par l'information obtenue lors des i premières étapes.

1.8.1 Algorithme séquentiellement optimal

Pour un $f \in F$ donné, soit $y_j = x_j(f)$, $x^i = (x_1, \ldots, x_i)$ et $y^i = (y_1, \ldots, y_i)$. L'information obtenue lors des i premières évaluations est $z^i = (x^i, y^i)$ et on définit

$$F(z^i) = \{f \in F \,|\, x_j(f) = y_j \;\; \forall j \leq i\}.$$

Si $F(z^i) \neq \emptyset$ alors on dit que z^i est une *situation réalisable.* Dans la définition suivante l'opération terminale ϕ est fixée et l'opérateur d'information est $N = (\tilde{x}_1, \ldots, \tilde{x}_n)$.

Définition 1.6 *Soit $\phi \in \hat{T}$ fixée. Pour chaque $i \leq n-1$ soit $\alpha_i = (\tilde{x}_{i+1}, \ldots, \tilde{x}_n, \phi)$*

un algorithme défini sur $F(z^i)$ pour tout z^i. Un algorithme $\alpha = (\alpha_1, \ldots, \alpha_n)$ est séquentiellement optimal (SO) si

1. *α est optimal sur F.*
2. *chaque α_i est optimal sur $F(z^i)$ pour toute situation réalisable z^i.*

Un algorithme SO est non seulement optimal mais aussi optimal pour chacun des problèmes définis successivement par l'information recueillie à chaque étape. Ceci permet à l'algorithme d'utiliser de la meilleure façon possible l'information favorable qui pourrait être obtenue au cours de son application à un problème donné. Même lorsque l'adapation n'est pas utile pour un problème, un algorithme SO donne généralement de meilleurs résultats qu'un algorithme optimal puisque celui-ci se prémunit seulement contre le pire cas, qui ne survient pas nécessairement. Cependant, un algorithme SO est plus difficile à mettre en oeuvre et possède généralement une complexité combinatoire élevée.

Exemple: Pour les problèmes d'approximation L^p et L_h^1 une situation réalisable est donnée par

$$z^i = (x_1, \ldots, x_i, f(x_1), \ldots, f(x_i), f'(x_1), \ldots, f'(x_i))$$

pour un $f \in F_{a,b}$ et $F_{a,b}(z^i)$ est l'ensemble des fonction de $F_{a,b}$ ayant les mêmes valeurs et dérivées que f aux points x_j. Un algorithme SO pour ces problèmes choisira le prochain point d'évaluation x_{i+1} en tenant compte des valeurs $f(x_j)$ et $f'(x_j)$ pour $j \leq i$ en plus du fait qu'il reste encore $n - i$ évaluations à faire. Une description détaillée d'un algorithme SO dans ce contexte est donnée à la section 6.1.1.

1.8.2 Algorithme optimal à chaque étape

Un algorithme séquentiellement optimal est souvent difficile à réaliser et coûteux du point vue calculatoire. Un algorithme optimal à chaque étape (*one-step optimal*, OSO) est plus simple et peut être efficace. Essentiellement un algorithme OSO garantit la plus grande amélioration à chaque étape, mais sans prendre en compte le nombre d'évaluations restantes.

Définition 1.7 *Pour $\phi \in \hat{T}$ fixé, un algorithme $\alpha^0 = (\tilde{x}_1^0, \ldots, \tilde{x}_n^0, \phi)$ est optimal à chaque étape si pour tout i*

$$\sup_{f \in F(z^i)} e((\tilde{x}_i^0, \phi), f) = \inf_{\tilde{x} \in \tilde{X}_{i+1}} \sup_{f \in F(z^i)} e((\tilde{x}, \phi), f)$$

pour toute situation réalisable $F(z^i)$.

Dans cette définition $\tilde{X}_{i+1}$ désigne un ensemble d'applications $X_1 \times \cdots \times X_i \times Y_1 \times \cdots \times Y_i \rightarrow X_{i+1}$.

Exemple: Pour nos problèmes d'approximation, un algorithme OSO choisira à chaque étape le prochain point d'évaluation x_{i+1} qui réduit au maximum l'erreur, en tenant compte des valeurs de $f(x_j)$ et $f'(x_j)$ déjà calculées. Une description détaillée d'un algorithme OSO dans ce contexte est donnée à la section 6.1.2.

1.8.3 Utilité pratique

Le développement et la mise en oeuvre d'un algorithme séquentiellement optimal est généralement plus difficile que celui d'un algorithme optimal dans le pire cas. En pratique l'amélioration de la précision apportée par un tel algorithme pourrait être

compensée par la plus grande complexité des calculs nécessaires. On pourrait cependant, connaissant un algorithme séquentiellement optimal, essayer de définir un algorithme qui en serait «proche» et dont la complexité (combinatoire et/ou informationnelle) serait moindre. Il s'agit donc, connaissant l'algorithme idéal en théorie, de l'utiliser pour juger de l'efficacité d'une approximation ou d'une heuristique. C'est ce qui est fait au chapitre 7 pour les problèmes d'approximation L^p et L_h^1, où des algorithmes optimaux, OSO et SO sont comparés.

CHAPITRE 2

REVUE DE LITTÉRATURE

Le point de vue de la complexité informationnelle pour les problèmes d'approximation a pris de l'importance à partir des années 50, motivé par le développement des premiers ordinateurs. Parmi les précuseurs on compte Nikolskij en 1950, qui a étudié le problème d'intégration et Kiefer [17] en 1957, qui a examiné des algorithmes séquentiels pour divers problèmes. Plus tard, Bakhvalov en 1971 a considéré le problème d'adaptation pour l'approximation d'opérateurs. À la même époque, les travaux de Traub et Woźniaskowski dans les années 60 et 70 mettaient l'emphase sur le rôle de l'information dans les problèmes d'approximation. Pour plus de détails et des références sur les les travaux cités ci-dessus, nous suggérons au lecteur la bibliographie dans Traub et Werschulz [44].

Selon ces derniers, l'histoire de la complexité informationnelle comporte trois phases. Dans la première les chercheurs se sont penchés sur des problèmes particuliers : intégration, reconstruction de fonctions, résolution d'équations différentielles, etc. Dans les phases suivantes, la théorie est systématisée : lors de la seconde pour les problèmes dans le contexte du pire cas et lors de la troisième pour les contextes stochastiques et moyens. Les textes de Traub, Wasilkowski et Wośsniakowski [41], [42] et [43] exposent de façon détaillée les concepts et résultats de la complexité informationnelle. La théorie générale propose des bornes sur l'erreur pour des classes de problèmes et des résultats sur l'utilité de l'adaptation, comme par exemple la proposition 1.3 de la section 1.7.3. Novak ([24, 27]) utilise avec profit les relations entre les concepts de n-diamètre et de rayon d'information pour obtenir des résultats généraux de même que des bornes sur l'erreur pour des problèmes particuliers.

Parallèlement à ces développments, Sukharev ([36, 38, 40]) élabore le concept d'optimalité séquentielle. Cette idée avait aussi été examinée par Sonnevend [32] dans le cas particulier de l'approximation dans la norme L^1 pour les fonctions de la classe $F_{a,b}$. Plus récemment, Korneichuk ([18, 19]) reprend le concept d'algorithme optimal à chaque étape du point de vue de la théorie des n-diamètres.

La théorie de même que les applications de la complexité informationnelle continuent à se développer. Parmi les sujets d'intérêt on compte les problèmes d'intégration en grande dimension (par exemple Hickernell et al. [16], Woźniakowski [46]) et les méthodes stochastiques pour ces problèmes (par exemple, Wang [48]).

Nous présentons maintenant une revue de la littérature concernant nos problèmes particuliers. Il est important de rappeler que toutes les données d'un problème : classe et opérateur étudiés, mesure de l'erreur et information disponible sont cruciales pour sa solution. Un petit changement peut entraîner une différence au niveau de la solution elle-même, de la difficulté du problème et de l'utilité de l'adaptation. Il importe donc de souligner ici les caractéristiques de nos problèmes, de façon à pouvoir faire des comparaisons avec d'autres travaux.

D'abord, comme mentionné en introduction, notre approche est pratique et nous nous efforçons le plus possible de donner les formules explicites à la fois pour les points d'évaluation optimaux et pour les bornes sur les erreurs. Ensuite, la classe $F_{a,b}$ que nous étudions est particulière : beaucoup d'information a priori est connue sur les fonctions à approximer, et on cherche à faire le meilleur usage de toute l'information disponible. Cependant elle est aussi assez générale au sens où il n'y a pas de restrictions sur, par exemple, la dérivabilité des fonctions. Ceci fait en sorte que plusieurs outils d'analyse ne nous sont pas accessibles.

Approximation L^1

Notre problème d'approximation dans la norme L^1 est équivalent au problème d'intégration, essentiellement parce qu'il existe pour chaque $F_{a,b}(z^n)$ une fonction maximale U et minimale L (voir la section 1.8.1 pour la notation). En fait, par un théorème de Sukharev [37], le problème est équivalent à trouver une formule de quadrature affine optimale.

Ce problème a été étudié par Glinkin [12] pour une classe de fonctions qui ne diffère de $F_{a,b}$ que par le fait que la valeur des fonctions n'est pas fixée aux extrémités de l'intervalle de définition. L'information utilisée est la valeur de la fonction seulement et on cherche à déterminer les points d'évaluation et la formule de quadrature linéaire optimaux. Des formules explicites sont données et la borne sur l'erreur trouvée est d'ordre $\mathcal{O}(n^{-2})$, où n est le nombre de points d'évaluation.

Le même problème est considéré par Zwick [47] qui trouve une formule affine optimale et améliore la borne sur l'erreur de [12], qui reste cependant du même ordre. L'algorithme déterminé par cette formule est optimal parmi les algorithmes passifs. En fait, il est démontré dans Novak [26] que l'adaptation n'est pas utile pour ce problème et donc la formule de Zwick est d'ordre optimal parmi tous les algorithmes. De plus, même lorsque l'information sur la dérivée est disponible l'adaptation n'est pas utile. Cette information supplémentaire peut donc réduire l'erreur d'approximation mais pas assez cependant pour en changer l'ordre par rapport à n.

Le problème d'approximation L^1 sur la classe $F_{a,b}$, avec l'information de f et f', est résolu par Sonnevend [33]. L'auteur montre d'abord que l'adaptation n'est pas utile au sens strict pour ce problème en exhibant la pire fonction et en utilisant le corollaire 1.1 de la section 1.7.3. Il donne ensuite des formules explicites pour les points d'évaluations optimaux et pour l'erreur d'approximation. Même si l'adaptation n'est pas utile pour ce problème un algorithme séquentiel est proposé. Il possède

l'avantage par rapport à la méthode passive de pouvoir être prolongé indéfiniment, c'est-à-dire que le nombre d'évaluations n'est pas fixé à l'avance. Cet algorithme est essentiellement l'algorithme du sandwich de Burkard, Hamacher et Rote [5], décrit à la section 4.3.

Les mêmes formules ont été retrouvées par Guérin, Marcotte et Savard [14], qui construisent un nouvel algorithme adaptatif de faible complexité combinatoire basé sur la programmation dynamique.

Approximation L^∞

Le problème d'approximation L^∞, aussi nommé problème de reconstruction optimale, a été étudié par plusieurs auteurs et pour diverses classes de fonctions. Nous présentons ici les travaux les plus pertinents pour nous.

Sonnevend [33] mentionne sans cependant le démontrer que l'algorithme adaptatif qu'il propose pour l'approximation L^1 s'applique aussi au cas L^∞ et donne une erreur d'ordre $\mathcal{O}(n^{-2})$, et mentionne aussi que pour ce problème l'adaptation est utile. Ces résultats sont démontrés et généralisés dans Sonnevend [35] lorsque seule l'évaluation de la fonction est permise mais, comme avant, les résultats s'appliquent aussi au cas où la dérivée est disponible. Un algorithme séquentiel d'ordre $\mathcal{O}(n^{-2})$ est proposé et il est démontré que cette borne est optimale quant à l'ordre. Pour montrer que l'adaptation est utile, on démontre que les algorithmes passifs ne peuvent pas faire mieux qu'une erreur d'ordre $\mathcal{O}(n^{-1})$. Ces bornes ne sont pas données explicitement cependant.

Burkard *et al.* [5] considèrent aussi le problème d'approximation L^∞ sur $F_{a,b}$. Leur point de vue n'est pas celui de la complexité informationnelle mais leur approche constructive leur permet de définir de façon simple et concrète un algorithme adaptatif et de prouver une borne explicite pour l'erreur de cet algorithme. On propose

en fait deux variantes de l'agorithme du sandwich de la section 4.3, qui diffèrent par le choix du point de subdivision à chaque étape. D'autres variantes sont décrites et comparées numériquement par Rote [30]. L'auteur conclut qu'aucune d'entre elles variantes étudiées n'est systématiquement meilleure que les autres et pose la question de savoir si un algorithme optimal (au sens de la section 1.6) peut être trouvé pour l'approximation L^∞. Il remarque que le contexte approprié pour y répondre est celui de la complexité informationnelle mais que le problème semble difficile : même si seulement deux évaluations sont à faire on ne sait pas comment déterminer les deux points optimaux. Yang et Goh [49] utilisent aussi l'algorithme du sandwich pour approximer des fonctions définies sur un intervalle fini dans le cas où la dérivée n'est pas disponible. Cependant leur méthode suppose que l'on connaisse (ou calcule) la fonction à approximer en tout point de [0,1].

Approximation L^p et L_h^1

La littérature sur les problèmes d'approximation L^p pour $1 < p < \infty$ et L^1 pondérée est beaucoup moins abondante que pour les problèmes d'intégration et de reconstruction optimale. En fait à notre connaissance ces problèmes n'ont pas été abordés pour la classe $F_{a,b}$ qui nous intéresse.

Le problème d'approximation L_h^1 étant équivalent au problème d'intégration pondérée, il a fait l'objet de plusieurs travaux mais souvent dans le cadre de l'approximation classique plutôt que dans celui de la complexité informationnelle. Dans ce cas les algorithmes proposés sont passifs et des restrictions sont imposées au type de formule/algorithme, ainsi qu'aux classes de fonctions, à l'information utilisée et au type de fonction de poids. Par exemple, Mathé [23] cherche une formule de quadrature pondérée pour des fonctions de Hölder sur la droite réelle et donne des conditions sur la fonction de poids pour que l'intégrale soit finie. Cerone, Roumeliotis et Hanna [6] proposent des formules de quadrature avec trois termes (trois évaluations) pour

des fonctions sur un intervalle fini et une fonction de poids générale. Leur approche calculatoire se rapproche de la nôtre mais leur point de vue n'est pas celui de la complexité informationnelle et ils ne considèrent pas le problème de déterminer si leur formule est en fait un algorithme optimal tel que défini au chapitre 1. Lee [20] étudie l'approximation pondérée dans la norme L^p dans le cadre de la théorie des ondelettes.

Lorsque le point de vue de la complexité informationnelle est adopté, les fonctions considérées sont souvent restreintes aux espaces classiques de l'analyse mathématique, des restrictions étant imposées sur les dérivées, et les fonctions de poids sont d'un type particulier. D'autres auteurs envisagent le problème de déterminer des caractéristiques de la fonction de poids qui rendent un certain problème plus simple. Par exemple, Curbera [7] montre que certaines fonctions de poids permettent de retarder la «malédiction de la dimension» (curse of dimension) pour l'intégration à plusieurs variables. Wasilkowski et Woźniaskowski [45] quant à eux cherchent à caractériser les fonctions de poids qui rendent finie la complexité du problème d'approximation L^p pondérée sur toute la droite réelle, autrement dit telles que le nombre de calculs d'information pour atteindre une borne donnée sur l'erreur soit fini.

Pour ce qui est de l'approximation dans la norme L^p, nous avons constaté que ce problème est souvent étudié en conjonction avec l'approximation pondérée, comme dans les travaux cités plus haut. Nous ne connaissons pas de travaux portant sur l'approximation L^p sur la classe $F_{a,b}$.

CHAPITRE 3

APPROXIMATION L^1 : RÉSUMÉ ET EXTENSIONS

Dans ce chapitre nous développons en détails l'étude de l'approximation des fonctions de $F_{a,b}$ dans la norme L^1. Après avoir démontré certains résultats généraux nous décrivons et généralisons le travail présenté dans Guérin *et al.* [14].

3.1 Résultats généraux

Nous démontrons ici les résultats annoncés aux sections 1.4 et 1.5 concernant l'opérateur d'information et l'opération terminale pour nos problèmes. Ces résultats s'appliquent non seulement à l'approximation L^1 mais aussi à l'approximation dans les normes L^p et L_h^1.

Rappelons que l'information sur les fonctions $f \in F_{a,b}$ à approximer est obtenue en évaluant f et f' en n points de [0,1]. Dans la suite, nous identifierons donc l'opérateur d'information N avec un n-tuplet $(x_1, \ldots, x_n)$ et nous dénoterons $x = (x_1, \ldots, x_n)$. Si on considère seulement des algorithmes passifs cette identification est tout à fait appropriée. Pour les algorithmes adaptatifs, le n-tuplet ne met pas en évidence les fonctionnelles $\tilde{x}_i$ qui déterminent les points x_i en fonction de l'information recueillie lors des évaluations précédentes. Cependant, ce qui nous intéresse dans ce qui suit est le résultat de l'application des $\tilde{x}_i$, c'est-à-dire les points d'évaluation. C'est pourquoi même dans ce cas N sera identifié à x, sous-entendu que ces points sont possiblement obtenus de façon adaptative.

Puisque $F_{a,b}$ contient des fonctions qui ne sont pas dérivables, il faut inclure dans

la définition de N, pour chaque $f \in F_{a,b}$ et $x \in [0,1]$, le choix d'un surgradient. Dénotons par $df(x)$ le surdifférentiel de f en x (c'est-à-dire l'ensemble de tous les surgradients de f en x). Un choix d'un surgradient est défini par un ensemble de fonctions $s = \{s_f\}_{f \in F_{a,b}}$ qui donnent pour $x \in [0,1]$ un élément $s_f(x) \in df(x)$. Un opérateur d'information peut maintenant être défini par

$$N(f) = (x_1, \ldots, x_n, f(x_1), \ldots, f(x_n), s_f(x_1), \ldots, s_f(x_n))$$

et on notera $N = (x, s)$.

Lemme 3.1 *Si N_1 et N_2 sont des opérateurs d'information qui ne diffèrent que par le choix des surgradients, c'est-à-dire que $N_1 = (x, s^1)$ et $N_2 = (x, s^2)$, alors $r(N_1) = r(N_2)$.*

Preuve: Soit $\bar{F}_{a,b}$ le sous-ensemble de $F_{a,b}$ constitué des fonctions dérivables. Pour tout $f \in F_{a,b}$ et $k = 1, 2$ il existe $\bar{f}_k \in \bar{F}_{a,b}$ avec $s_f^k(x_i) = \bar{f}_k'(x_i)$ pour tout $i = 1, \ldots, n$. Dans ce cas les fonctions U et L construites à partir de l'information $f(x_i)$, $s_f^k(x_i)$ et $\bar{f}_k(x_i)$, $\bar{f}_k'(x_i)$ sont les mêmes. On peut donc calculer le rayon d'information de N_1 et N_2 en considérant seulement les fonctions de $\bar{F}_{a,b}$. Or pour ces fonctions l'information donnée par les deux opérateurs N_k est la même donc

$$r(N_1) = \sup_{f \in \bar{F}_{a,b}} ||U(x,f) - L(x,f)|| = r(N_2).$$

∎

Ce lemme signifie qu'il n'y a pas de meilleur choix pour les surgradients.

Lemme 3.2 *Si $\alpha_i = (N_i, \phi)$, pour $i = 1, 2$, et que N_1 et N_2 ne diffèrent que par le choix des surgradients alors $e(\alpha_1) = e(\alpha_2)$.*

Preuve: Comme dans la preuve du lemme 3.1 on peut calculer $e(\alpha_1)$ et $e(\alpha_2)$ en considérant seulement les fonctions dérivables. Pour ces fonctions, $\phi(N_1(f)) = \phi(N_2(f))$ et on doit donc avoir $e(\alpha_1) = e(\alpha_2)$. ∎

Ce lemme implique que quelque soit le choix des sous-gradients, la solution au problème de minimiser $e(\alpha)$ est la même.

L'opération terminale ϕ^* a été définie à la section 1.5 par $\phi^* = \frac{1}{2}(U + L) \in F_{a,b}$, où U et L dépendent de x et f (voir aussi la section 3.2). Lorsqu'il sera nécessaire de rendre explicite cette dépendance on écrira $U \equiv U(x, f)$ et $L \equiv L(x, f)$. Lorsque de plus on voudra rendre explicite la variable d'intégration t, on notera $U \equiv U(x, f, t)$ et $L \equiv L(x, f, t)$.

Proposition 3.1 *L'opération terminale ϕ^* est centrale.*

Preuve: Pour le problème d'approximation, $S(f) = f$ donc $W(f) = V(f)$. Il faut montrer que pour tout x et f

$$\sup_{\tilde{f} \in V(f)} \left|\left| \frac{1}{2}(U(x, f) + L(x, f)) - \tilde{f} \right|\right| = \inf_{g \in F_{a,b}} \sup_{\tilde{f} \in V(f)} ||\, g - \tilde{f}\, ||. \tag{3.1}$$

Soit x et f fixés. Par définition de U et L, $0 \leq L \leq \tilde{f} \leq U$ pour tout $\tilde{f} \in V(f)$, ce qui implique que

$$-\frac{U - L}{2} \leq \frac{1}{2}(U + L) - \tilde{f} \leq \frac{U - L}{2}.$$

Donc puisque $U \in V(f)$,

$$\sup_{\tilde{f} \in V(f)} \left|\left| \frac{1}{2}(U + L) - \tilde{f} \right|\right| = \frac{1}{2}||U - L||. \tag{3.2}$$

Maintenant soit $g \in F_{a,b}$ et montrons que

$$\sup_{\tilde{f} \in V(f)} || g - \tilde{f} || \geq \frac{1}{2} ||U - L||. \tag{3.3}$$

Si $||g - L|| > \frac{1}{2}||U - L||$ alors (3.3) est satisfaite car il existe des fonctions de $V(f)$ qui sont arbitrairement proches de L. Sinon

$$\begin{aligned} ||U - L|| &\leq ||U - g|| + ||g - L|| \\ &\leq ||U - g|| + \frac{1}{2}||U - L||, \end{aligned}$$

ce qui implique $\frac{1}{2}||U - L|| \leq ||g - U||$ et (3.3) est satisfaite car $U \in V(f)$. Finalement, (3.3) et (3.2) impliquent (3.1) car $\frac{1}{2}(U + L) \in F_{a,b}$, donc ϕ^* est bien un centre de $V(f)$. ∎

Comme corollaires de la preuve de la proposition 3.1 on a les expressions suivantes pour le rayon d'information et l'erreur locale :

Corollaire 3.1 *Pour ϕ^* fixé et $N = (x_1, \ldots, x_n)$, on a*

$$r(N) = \frac{1}{2} \sup_{f \in F_{a,b}} ||U(x, f) - L(x, f)||.$$

Preuve: Par la définition du rayon d'information et les égalités (3.1) et (3.2) de la preuve de la proposition 3.1 on a

$$\begin{aligned} r(N) &= \sup_{f \in F_{a,b}} \left(\inf_{g \in F_{a,b}} \sup_{\tilde{f} \in V(f)} ||g - \tilde{f}|| \right) \\ &= \frac{1}{2} \sup_{f \in F_{a,b}} ||U(x, f) - L(x, f)||. \end{aligned}$$

∎

Corollaire 3.2 *Pour un algorithme $\alpha = (x_1, \ldots, x_n, \phi^*)$ l'erreur locale en f est*

$$e_{loc}(\alpha, f) = \frac{1}{2}||U(x, f) - L(x, f)||.$$

Preuve: Par la définition de e_{loc} et l'égalité (3.2) de la preuve de la proposition 3.1 on a

$$\begin{aligned} e_{loc}(\alpha, f) &= \sup_{\tilde{f} \in V(f)} || \alpha(f) - \tilde{f} || \\ &= \sup_{\tilde{f} \in V(f)} \left|\left| \frac{1}{2}(U(x, f) + L(x, f)) - \tilde{f} \right|\right| \\ &= \frac{1}{2} || U(x, f) - L(x, f) || . \end{aligned}$$

∎

3.2 Approximation L^1

Rappelons brièvement les données de ce problème. Soit

$$F_{a,b} = \{f : [0,1] \to \mathbf{R} \mid f \text{ concave }, f(0) = 0, f(1) = 1, f'(0) \leq a, f'(1) \geq b\}$$

où $f'(0)$ et $f'(1)$ sont les dérivées à gauche et à droite en $x = 0$ et $x = 1$, respectivement. On veut approximer les fonctions de $F_{a,b}$ à l'aide d'un algorithme $\alpha = (N, \phi)$, où l'opérateur d'information N est identifié avec un vecteur $x = (x_1, \ldots, x_n)$ représentant les points d'évaluation, comme à la section 3.1, et l'opération terminale est $\phi^* = \frac{1}{2}(U + L)$ qui est centrale et donc optimale.

Explicitement, les fonctions U et L sont définies comme suit. Posons $x_0 = 0$, $x_{n+1} =$

1, $v_i = f(x_i)$, $\mu_i = f'(x_i)$ et $v = (v_0, \ldots, v_{n+1})$, $\mu = (\mu_0, \ldots, \mu_{n+1})$. Alors

$$U(x, f, t) \equiv U(x, v, \mu, t) = \min_{0 \le i \le n+1} \{\mu_i(t - x_i) + v_i\}$$

$$L(x, f, t) \equiv L(x, v, \mu, t) = \min_{0 \le i \le n} \left\{ \left(\frac{v_{i+1} - v_i}{x_{i+1} - x_i} \right) (t - x_i) + v_i \right\}.$$

Puisque les fonctions de $F_{a,b}$ sont concaves, v et μ doivent satisfaire

$$C(x) : \begin{cases} \dfrac{v_{i+1} - v_i}{x_{i+1} - x_i} \le \mu_i \le \dfrac{v_i - v_{i-1}}{x_i - x_{i-1}}, \quad i = 1, \ldots, n \\ \\ x_0 = 0,\ x_{n+1} = 1,\ v_0 = 0,\ v_{n+1} = 1,\ \mu_0 = a,\ \mu_{n+1} = b. \end{cases} \tag{3.4}$$

Pour x fixé, ces conditions seront notées $(v, \mu) \in C(x)$.

Pour le problème d'approximation L^1 Sonnevend [33] a montré que l'adaptation n'est pas utile au sens strict en construisant explicitement une fonction pire cas et en utilisant le corollaire 1.1. Ce résultat est également conséquence de la proposition 4.3 du chapitre 4.

Puisque de plus ϕ^* est fixée, le problème de trouver un algorithme optimal se réduit à trouver les points d'évaluation optimaux. Le problème s'énonce maintenant comme suit :

$$\begin{aligned} \min_{x \in [0,1]^n} e(x) &= \min_{x \in [0,1]^n} \sup_{f \in F_{a,b}} e_{loc}(x, f) \\ &= \min_{x \in [0,1]^n} \sup_{(v,\mu) \in C(x)} \frac{1}{2} \int_0^1 [U(x, v, \mu, t) - L(x, v, \mu, t)]\, dt. \end{aligned} \tag{3.5}$$

Ici $e(x)$ est l'erreur de l'algorithme (x, ϕ^*).

3.3 Formulation de programmation dynamique

Dans Guérin *et al.* [14] une formulation de programmation dynamique, inspirée d'un problème semblable de Bellman et Dreyfus [3], a été employée pour trouver une solution au problème (3.5) restreint aux fonctions croissantes de $F_{a,b}$. Nous rappelons cette formulation avant d'en donner une généralisation à toutes les fonctions de $F_{a,b}$.

Soit f une fonction concave avec $f(x_1) = v_1$, $f(x_2) = v_2$, $f'(x_1) = \mu_1$ et $f'(x_2) = \mu_2$. Définissons un changement de variable par

$$T(t) = \frac{t - x_1}{x_2 - x_1}$$

$$S(s) = \frac{s - v_1}{v_2 - v_1}$$

et une nouvelle fonction

$$\bar{f}(t) = S \circ f \circ T^{-1}(t).$$

La fonction $\bar{f}$ est définie sur [0,1] et satisfait $f(0) = 0$ et $f(1) = 1$. Si $v_1 < v_2$ alors $\bar{f}$ est concave. Ceci est le cas, en particulier, si f est croissante. Sinon, $\bar{f}$ est *convexe*. Cependant, le problème d'approximation L^1 pour les fonctions convexes est essentiellement le même que pour les fonctions concaves. Définissons donc

$$\hat{F}_{a,b} = \{f : [0,1] \to \mathbf{R} \mid f \text{ convexe }, f(0) = 0, f(1) = 1, f'(0) \geq a, f'(1) \leq b\}$$

et

$$E_1(n)(a,b) = \begin{cases} \text{erreur minimale dans le pire cas pour l'approximation de} \\ \text{fonctions de } F_{a,b}, \text{ si } a \geq 1 \text{ et qu'il y a } n \text{ évaluations permises,} \\ \\ \text{erreur minimale dans le pire cas pour l'approximation de} \\ \text{fonctions de } \hat{F}_{a,b}, \text{ si } a < 1 \text{ et qu'il y a } n \text{ évaluations permises.} \end{cases}$$

Maintenant, pour établir une relation de récurrence pour l'erreur $E_1(n)$ on a

$$\int_{x_1}^{x_2} [U(t) - L(t)]\, dt = (x_2 - x_1)(v_2 - v_1) \int_0^1 [\bar{U}(t) - \bar{L}(t)]\, dt, \tag{3.6}$$

où

$$U(t) = \min\{\mu_1(t - x_1) + v_1, \mu_2(t - x_2) + v_2\}$$

$$L(t) = \left(\frac{v_2 - v_1}{x_2 - x_1}\right)(t - x_1) + v_1.$$

Il est facile de montrer que la formule (3.6) est valide si $v_1 < v_2$ ou $v_1 > v_2$. En fait, en posant $v_2 = v_1 + \epsilon$ et en prenant la limite quand $\epsilon \to 0$ on peut montrer qu'elle est aussi valide lorsque $v_1 = v_2$. Enfin, on a

$$E_1(n)(a, b) =$$

$$\min_{x \in [0,1]} \max_{v,\mu \in C(x)} \left\{ xvE_1(0)\left(\frac{x}{v}a, \frac{x}{v}\mu\right) + (1 - x)(1 - v)E_1(n - 1)\left(\frac{1 - x}{1 - v}\mu, \frac{1 - x}{1 - v}b\right)\right\}. \tag{3.7}$$

Le point x est le premier point d'approximation sur $[0, 1]$ et, dans la somme entre accolades, le terme de gauche représente l'erreur sur $[0, x]$ tandis que celui de droite représente l'erreur sur $[x, 1]$ étant donné que les $n-1$ autres évaluations seront faites sur cet intervalle.

Le point x^* qui minimise l'expression (3.7) est

$$x^* = \frac{1}{(n+1)^2}\left(1 + 2n\frac{1 - b}{a - b}\right) \tag{3.8}$$

et l'erreur optimale est

$$E_1(n)(a, b) = \frac{1}{2(n+1)^2}\frac{(a - 1)(1 - b)}{(a - b)}. \tag{3.9}$$

Pour plus de détails concernant cette formulation, voir Guérin [15].

La relation de récurrence (3.7) avec la formule (3.8) donnent une méthode adaptative pour calculer successivement de gauche à droite sur [0,1] les points d'évaluation optimaux. Puisque l'adaptation n'est pas utile au sens strict pour ce problème, cette méthode ne peut pas faire mieux dans le pire cas que l'algorithme passif optimal proposé par Sonnevend [33]. Cependant l'algorithme obtenu par cette méthode est très simple, sa complexité combinatoire étant la même que celle de l'algorithme passif. Il est en pratique plus efficace que ce dernier, comme le montrent les résultats numériques du chapitre 7.

Finalement, pour l'approximation de fonctions avec $f(x_1) = v_1$, $f(x_2) = v_2$, $f'(x_1) = \mu_1$ et $f'(x_2) = \mu_2$ on a la formule générale

$$x^* = \frac{x_2 - x_1}{(n+1)^2}\left[1 + 2n\left(\frac{v_2 - v_1 - (x_2 - x_1)\mu_2}{(x_2 - x_1)(\mu_1 - \mu_2)}\right)\right] + x_1 \tag{3.10}$$

pour le premier point d'évaluation. L'erreur minimale dans ce cas est

$$\frac{1}{2(n+1)^2}\frac{[\mu_1(x_1 - x_1) - (v_2 - v_1)][v_2 - v_1 - \mu_2(x_2 - x_1)]}{\mu_1 - \mu_2}. \tag{3.11}$$

CHAPITRE 4

APPROXIMATION L^p

Nous examinons dans cette section le problème d'approximation dans la norme L^p pour $1 \leq p \leq \infty$, qui généralise la section précédente. Les deux principaux résultats de cette section sont la proposition 4.1 qui donne une borne inférieure sur l'erreur pour les algorithmes passifs et la proposition 4.2 qui donne une borne supérieure pour l'erreur de l'algorithme du sandwich, qui est un algorithme adaptatif d'ordre optimal. Ces deux propositions impliquent que l'adaptation est utile pour ce problème si et seulement si $p > 1$.

4.1 Préliminaires

Les données du problèmes sont les mêmes qu'au chapitre 3 sauf pour l'erreur d'approximation qui est maintenant mesurée dans la norme L^p pour $1 \leq p \leq \infty$: si $f, g \in F_{a,b}$ alors

$$\begin{cases} \| f - g \|_p = \left(\int_0^1 | f(t) - g(t) |^p \, dt \right)^{\frac{1}{p}} & \text{si } p < \infty, \\ \| f - g \|_\infty = \max\limits_{t \in [0,1]} | f(t) - g(t) | & \text{si } p = \infty. \end{cases}$$

Les résultats de la section 3.1 concernant l'opérateur d'information et l'opération terminale $\phi^* = \frac{1}{2}(U + L)$ sont encore valables ici. En particulier, ϕ^* est centrale et donc optimale.

Pour simplifier, nous dénoterons la fonction $\frac{1}{2}(U - L)$ par UL et nous utiliserons la notation v, μ du chapitre 3 pour la valeur d'une fonction et de son surgradient aux points d'évaluation.

Pour $p < \infty$ si les points $x = (x_1, \ldots, x_n)$ ont été choisis alors l'erreur d'approximation sur la fonction f est

$$e(x, f) = \left(\sum_{i=0}^{n} \int_{x_i}^{x_{i+1}} UL(x, v, \mu, t)^p \, dt \right)^{\frac{1}{p}}.$$

Puisque $UL \geq 0$, $e(x, f)$ est plus grande ou égale à l'erreur sur un sous-intervalle donné $[x_i, x_{i+1}]$, qui est égale à

$$e_i = \left(\int_{x_i}^{x_{i+1}} UL(x, v, \mu, t)^p \, dt \right)^{\frac{1}{p}}.$$

Le lemme suivant donne la valeur exacte de e_i de même qu'une borne supérieure sur cette quantité. Cette borne sera utilisée dans la preuve de la proposition 4.2.

Lemme 4.1 *1. Sur l'intervalle* $[x_i, x_{i+1}]$, *si* $f(x_i) = v_i, f'(x_i) = \mu_i$, $f(x_{i+1}) = v_{i+1} et f'(x_{i+1}) = \mu_{i+1}$ *alors*

$$e_i = \frac{1}{2(1+p)^{\frac{1}{p}}} \frac{(x_{i+1} - x_i)^{1+\frac{1}{p}}}{\mu_i - \mu_{i+1}} \left[\mu_i - \frac{v_{i+1} - v_i}{x_{i+1} - x_i} \right] \left[\frac{v_{i+1} - v_i}{x_{i+1} - x_i} - \mu_{i+1} \right]. \tag{4.1}$$

2. L'erreur d'approximation (4.1) ci-dessus est bornée comme suit

$$e_i \leq \frac{(x_{i+1} - x_i)^{1+\frac{1}{p}} (\mu_i - \mu_{i+1})}{8(1+p)^{\frac{1}{p}}}. \tag{4.2}$$

Preuve:

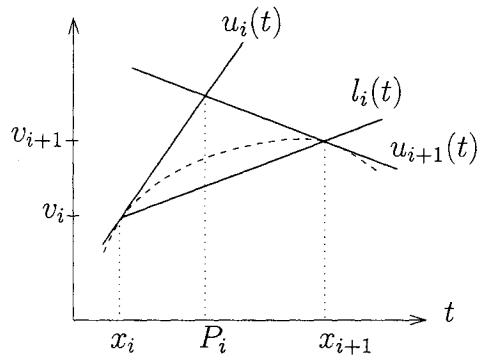


Figure 4.1 La situation sur $[x_i, x_{i+1}]$.

1. Soit P_i l'abcisse du point d'intersection des tangentes $u_i(t) = \mu_i(t - x_i) + v_i$ et $u_{i+1}(t) = \mu_{i+1}(t - x_{i+1}) + v_{i+1}$ en x_i et x_{i+1}, tel qu'illustré sur la figure 4.1. On a alors

$$P_i = \frac{\mu_i x_i - \mu_{i+1} x_{i+1} + v_{i+1} - v_i}{\mu_i - \mu_{i+1}}.$$

Sur cette figure est également illustrée

$$l_i(t) = \left(\frac{v_{i+1} - v_i}{x_{i+1} - x_i}\right)(t - x_i) + v_i = \left(\frac{v_{i+1} - v_i}{x_{i+1} - x_i}\right)(t - x_{i+1}) + v_{i+1}.$$

Pour $p < \infty$, la formule (4.1) est le résultat du calcul de

$$e_i = \frac{1}{2}\left(\int_{x_i}^{P_i} (u_i(t) - l_i(t))^p \, dt + \int_{P_i}^{x_{i+1}} (u_{i+1}(t) - l_i(t))^p \, dt\right)^{\frac{1}{p}}.$$

Pour $p = \infty$, il faut calculer

$$e_i = \max_{t \in [x_i, x_{i+1}]} UL(x, v, \mu, t) = \frac{1}{2}(u_i(P_i) - l_i(P_i)).$$

La formule (4.1) résulte de la simplification de ces expressions.

2. Posons $X = x_{i+1} - x_i$ et $V = v_{i+1} - v_i$. Pour X, μ_i et μ_{i+1} fixés, le maximum sur V de la fonction quadratique $(\mu_i - \frac{V}{X})(\frac{V}{X} - \mu_{i+1})$ est atteint lorsque $V = \frac{1}{2}(\mu_i + \mu_{i+1})X$. En substituant dans l'expression de l'erreur (4.1) on obtient

$$\begin{aligned} e_i &\leq \frac{X^{1+\frac{1}{p}}\left[\mu_i - \frac{1}{2}(\mu_i + \mu_{i+1})X\right]\left[\frac{1}{2}(\mu_i + \mu_{i+1})X - \mu_{i+1}\right]}{2(1+p)^{\frac{1}{p}}(\mu_i - \mu_{i+1})} \\ &= \frac{X^{1+\frac{1}{p}}(\mu_i - \mu_{i+1})}{8(1+p)^{\frac{1}{p}}}. \end{aligned}$$

Ceci complète la preuve du lemme. ∎

La borne (4.2) a l'avantage d'être linéaire en μ_i et μ_{i+1}, ce qui simplifiera certains calculs à la section suivante. Elle est aussi exacte au sens où il y a égalité pour certaines valeurs de x_i, v_i, μ_i. Cependant, la différence entre e_i et la borne devient arbitrairement grande lorsque $\mu_i \to \infty$ ou $\mu_{i+1} \to -\infty$.

4.2 Borne inférieure pour les algorithmes passifs

Nous calculons dans cette section une borne inférieure pour l'erreur des algorithmes passifs pour le problème d'approximation L^p.

Puisque l'opération terminale ϕ^* est fixée, un algorithme est déterminé par le choix des points d'évaluation $x = (x_1, \ldots, x_n)$. Pour un algorithme passif on peut considérer que tous les x_i sont choisis à l'avance en fonction seulement de l'information a priori donnée par $F_{a,b}$.

Proposition 4.1 *Soit α un algorithme passif pour le problème d'approximation L^p,*

avec $1 \leq p \leq \infty$, *sur la classe* $F_{a,b}$. *Alors*

$$e(\alpha) \geq \frac{1}{(n+1)^{1+\frac{1}{p}}} \left[\frac{(a-1)(1-b)}{12(1+p)^{\frac{1}{p}}(a-b)} \right].$$

Preuve: Si n points d'évaluation sont choisis, l'intervalle [0,1] est subdivisé en $n+1$ sous-intervalles et au moins un d'entre eux est de longueur $\delta \geq \frac{1}{n+1}$. Pour simplifier la notation, dénotons cet intervalle par $[x_1, x_2]$, où $x_2 = x_1 + \delta$. On peut supposer que $\delta = \frac{1}{n+1}$, quitte à se restreindre à un sous-intervalle de $[x_1, x_2]$. Nous allons montrer que l'on peut trouver une fonction $f \in F_{a,b}$ pour laquelle l'erreur est d'au moins la borne inférieure annoncée ci-dessus. Pour calculer l'erreur sur $[x_1, x_2]$, notée e, on substitue $v_1 = f(x_1), v_2 = f(x_2), \mu_1 = f'(x_1)$ et $\mu_2 = f'(x_2)$ dans la formule (4.1) du lemme 4.1.

Dans la situation représentée à la figure (4.1) lorsque $x_i = v_i = 0, \mu_i = a$, $x_{i+1} = v_{i+1} = 1$ et $\mu_{i+1} = b$ on retrouve la situation standard où aucun point n'a encore été placé sur [0,1]. Le point correspondant à P_i dans ce cas est $P = \frac{1-b}{a-b}$ et $u_i(t) = at$, $u_{i+1}(t) = bt + 1 - b$, $l_i(t) = t$.

Maintenant trois cas sont à considérer, dépendant de la position de x_1 et x_2.

Cas 1 : $0 \leq x_1 \leq P - \delta$. Tel qu'illustré à la figure 4.2, soit

$$l_1(t) = at, \quad v_1 = l_1(x_1),$$

$$l_2(t) = \left(\frac{1 - v_1}{1 - x_1} \right)(t-1) + 1, \quad v_2 = \frac{1}{2}(l_1(x_2) + l_2(x_2)),$$

$$l_3(t) = \left(\frac{v_2 - v_1}{x_2 - x_1} \right)(t - x_1) + v_1, \quad l_4(t) = \left(\frac{1 - v_2}{1 - x_2} \right)(t-1) + 1.$$

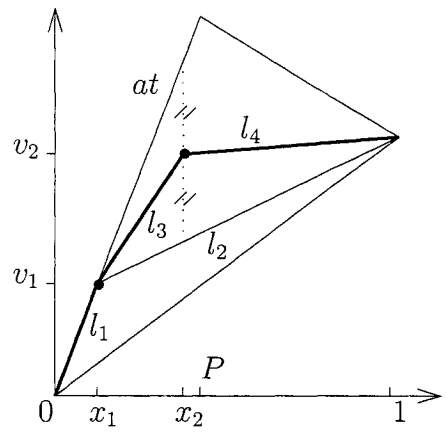


Figure 4.2 Proposition 4.1, cas 1.

Pour la fonction $f \in F_{a,b}$ définie par

$$f(t) = \begin{cases} l_1(t) & \text{si } 0 \leq t \leq x_1 \\ l_3(t) & \text{si } x_1 \leq t \leq x_2 \\ l_4(t) & \text{si } x_2 \leq t \leq 1 \end{cases}$$

on a $\mu_1 = f'(x_1) = a$ et $\mu_2 = f'(x_2) = \frac{1-v_2}{1-x_2}$. En substituant les valeurs de x_1, v_1, μ_1 et de x_2, v_2, μ_2 dans l'expression (4.1) du lemme 4.1 on trouve que l'erreur sur $[x_1, x_2]$ pour cette fonction est

$$e = \frac{1}{4} \frac{\delta^{1+\frac{1}{p}}}{(1+p)^{\frac{1}{p}}} \cdot \frac{a-1}{2(1-x_1)-\delta}.$$

Cette quantité est croissante en x_1 et atteint son minimum sur l'intervalle considéré en $x_1 = 0$. En substituant $x_1 = 0$ et $\delta = \frac{1}{n+1}$ on obtient

$$e \geq \frac{1}{4(1+p)^{\frac{1}{p}}} \cdot \frac{1}{(n+1)^{\frac{1}{p}}} \cdot \frac{a-1}{2n+1}$$

$$\geq \frac{1}{8(1+p)^{\frac{1}{p}}} \cdot \frac{(a-1)}{(n+1)^{1+\frac{1}{p}}}$$

$$\geq \frac{1}{(n+1)^{1+\frac{1}{p}}} \left[\frac{(a-1)(1-b)}{12(1+p)^{\frac{1}{p}}(a-b)} \right].$$

La dernière inégalité provient du fait que $\frac{1-b}{a-b} \leq 1$.

Cas 2 : $P - \delta \leq x_1 \leq P$. Tel qu'illustré à la figure 4.3, soit

$$v_1 = \frac{(a+1)x_1}{2}, \qquad l_1(t) = \frac{v_1}{x_1}t, \qquad l_2(t) = \left(\frac{1-v_1}{1-x_1}\right)(t-1)+1,$$

$$w = \frac{1}{2}(l_1(P) + l_2(P)), \qquad l_3(t) = \left(\frac{w-v_1}{P-x_1}\right)(t-x_1)+v_1.$$

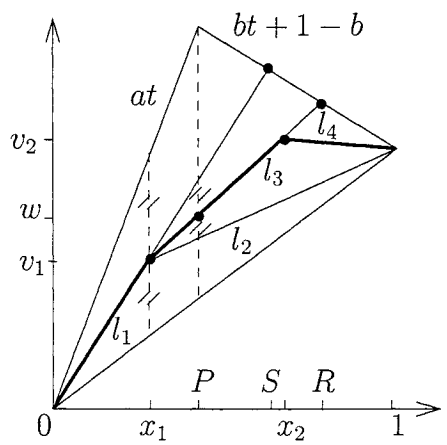


Figure 4.3 Proposition 4.1, cas 2, $x_2 \leq R$.

Soit aussi S et R les points tels que $l_1(S) = bS + 1 - b$ et $l_3(R) = bR + 1 - b$. On a $P \leq S \leq R$. Supposons d'abord que $x_2 \leq R$. Dans ce cas, soit

$$v_2 = l_3(x_2) \text{ et } l_4(t) = \left(\frac{1-v_2}{1-x_2}\right)(t-1)+1.$$

Pour la fonction $f \in F_{a,b}$ définie par

$$f(t) = \begin{cases} l_1(t) & \text{si } 0 \le t \le x_1 \\ l_3(t) & \text{si } x_1 \le t \le x_2 \\ l_4(t) & \text{si } x_2 \le t \le 1 \end{cases}$$

on a $\mu_1 = \frac{v_1}{x_1}$ et $\mu_2 = \frac{1-v_2}{1-x_2}$ et avec ces valeurs l'erreur pour f sur $[x_1, x_2]$ est

$$e = \frac{1}{4}\frac{\delta^{1+\frac{1}{p}}}{(1+p)^{\frac{1}{p}}} \cdot \frac{a-1}{2(1-x_1)-\delta}.$$

Comme avant, cette quantité est croissante en x_1 et atteint ici son minimum en $P-\delta$, ce qui donne en substituant cette valeur et celle de δ dans l'expression de l'erreur du lemme 4.1

$$\begin{aligned} e &\ge \frac{1}{4(1+p)^{\frac{1}{p}}} \cdot \frac{1}{(n+1)^{\frac{1}{p}}} \cdot \frac{(a-1)(a-b)}{2(a-1)(n+1)+a-b} \\ &\ge \frac{1}{4(1+p)^{\frac{1}{p}}} \cdot \frac{1}{(n+1)^{\frac{1}{p}}} \cdot \frac{(a-1)(a-b)}{(a-b)(2n+3)} \\ &\ge \frac{1}{(n+1)^{1+\frac{1}{p}}} \left[\frac{(a-1)(1-b)}{12(1+p)^{\frac{1}{p}}(a-b)}\right]. \end{aligned}$$

La deuxième inégalité ci-dessus vient du fait que $a-1 \le a-b$ et la troisième du fait que $\frac{1}{2n+3} \ge \frac{2}{5(n+1)}$ pour $n \ge 1$ et que $\frac{1-b}{a-b} \le 1$.

Supposons maintenant que $x_2 > R$ (voir figure 4.4). Alors

$$P-\delta \le S-\delta \le R-\delta < x_1 \le P \le S \le R.$$

Pour la fonction $f \in F_{a,b}$,

$$f(t) = \begin{cases} l_1(t) & \text{si } 0 \le t \le x_1 \\ l_3(t) & \text{si } x_1 \le t \le R \\ bt + 1 - b & \text{si } R \le t \le 1 \end{cases}$$

on a $v_2 = bx_2 + 1 - b$, $\mu_1 = \frac{v_1}{x_1}$ et $\mu_2 = b$. Avec ces valeurs, l'erreur pour f sur $[x_1, x_2]$ est

$$e = \frac{\delta^{1+\frac{1}{p}}}{4(1+p)^{\frac{1}{p}}} \cdot (a + 1 - 2b) \cdot \frac{1}{\delta^2}(S - x_1)(x_1 + \delta - S). \tag{4.3}$$

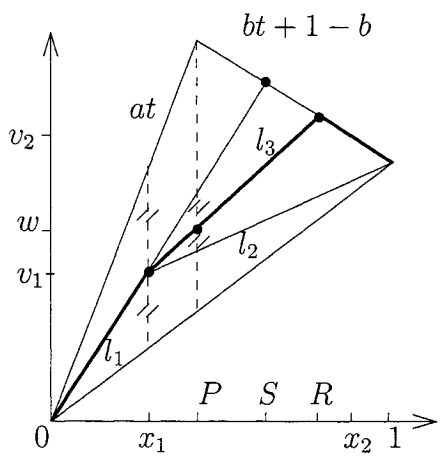


Figure 4.4 Proposition 4.1, cas 2, $x > R$.

Soit $g(x) = (S - x)(x + \delta - S)$. Cette fonction quadratique s'annule en S et en $S - \delta$, et son maximum est en $x^* = S - \frac{\delta}{2} \in]S - \delta, S[$.

• Si $x^* \le R - \delta \le P$ alors le minimum de g sur $[R - \delta, P]$ est en P et $g(P) = (S - P)(\delta - (S - P))$. Or $x^* \le P \Rightarrow \delta - (S - P) \ge \frac{\delta}{2}$ et $\delta = \frac{1}{n+1} \le \frac{1}{2}$ donc

$$\frac{1}{\delta^2} g(x_1) \ge \frac{1}{\delta^2} g(P) \ge \frac{1}{\delta^2}(S - P) \cdot \frac{\delta}{2} \ge S - P.$$

• Si $R-\delta \leq x^* \leq P$ alors le minimum de g sur $[R-\delta, P]$ est à l'une des extrémités de cet intervalle. S'il est en P, on a comme avant $\frac{1}{\delta^2}g(x_1) \geq S-P$. S'il est en $R-\delta$ alors $g(R-\delta) = (\delta-(R-S))(R-S)$. Or $x^* \geq R-\delta \Rightarrow \delta-(R-S) \geq \frac{\delta}{2}$ donc

$$\frac{1}{\delta^2}g(x_1) \geq \frac{1}{\delta^2}g(R-\delta) \geq \frac{1}{\delta^2}(R-S)\cdot\frac{\delta}{2} \geq R-S.$$

• Si $x^* \geq P$ alors le minimum de g sur $[R-\delta, P]$ est en $R-\delta$ et comme auparavant $\frac{1}{\delta^2}g(x_1) \geq R-S$.

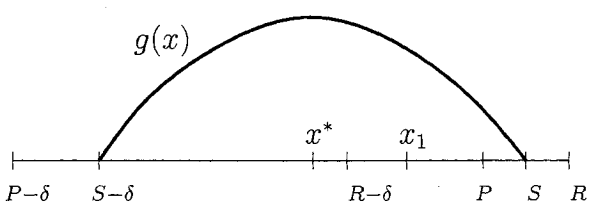


Figure 4.5 La fonction g.

On conclut donc que $\frac{1}{\delta^2}g(x_1) \geq \min\{S-P, R-S\}$. Si le minimum est $S-P$ alors, puisque $S = \frac{2(1-b)}{a+1-2b}$ et $S-P = \frac{(a-1)(1-b)}{(a+1-2b)(a-b)}$, on obtient en substituant ces valeurs dans (4.3)

$$\begin{aligned} e &\geq \frac{1}{4(1+p)^{\frac{1}{p}}}\cdot\delta^{1+\frac{1}{p}}\cdot\frac{(a-1)(1-b)}{a-b} \\ &\geq \frac{1}{(n+1)^{1+\frac{1}{p}}}\left[\frac{(a-1)(1-b)}{12(1+p)^{\frac{1}{p}}(a-b)}\right]. \end{aligned}$$

Si le minimum est en $R-S$ il faut condsidérer le point R, qui dépend de x_1 :

$$R \equiv R(x_1) = \frac{(a-b+3(1-b))x_1 - 4(1-b)}{2((a-b)+(1-b))x_1 - (a-b+3(1-b))}.$$

Cette fonction est décroissante en x_1 et atteint son minimum sur $[P-\delta, P]$ en P.

On a alors

$$R - S \geq R(P) - S = \frac{(a-1)(1-b)}{(a+1-2b)(a+2-3b)}.$$

Puisque $\frac{1}{a+2-3b} \geq \frac{1}{3(a-b)}$, en substituant dans (4.3) on trouve

$$\begin{aligned} e &\geq \frac{1}{4(1+p)^{\frac{1}{p}}} \cdot \delta^{1+\frac{1}{p}} \cdot \frac{(a-1)(1-b)}{3(a-b)} \\ &= \frac{1}{(n+1)^{1+\frac{1}{p}}} \left[\frac{(a-1)(1-b)}{12(1+p)^{\frac{1}{p}}(a-b)} \right]. \end{aligned}$$

Cas 3 : $x_1 \geq P$. Soit

$$l_1(t) = bt + 1 - b, \qquad v_2 = l_1(x_2), \qquad l_2(t) = \frac{v_2}{x_2}t, \qquad v_1 = \frac{1}{2}(l_1(x_1) + l_2(x_1)),$$

$$l_3(t) = \left(\frac{v_2 - v_1}{x_2 - x_1}\right)(t - x_2) + v_2, \qquad l_4(t) = \frac{v_1}{x_1}t$$

tel qu'illustré à la figure 4.6.

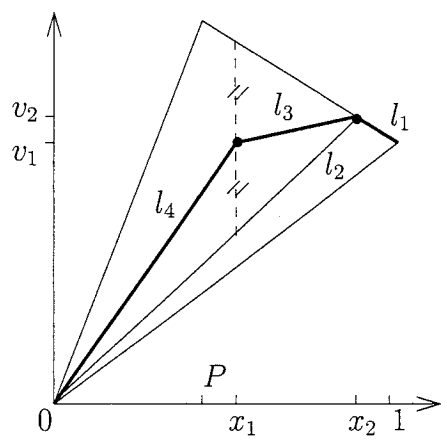


Figure 4.6 Proposition 4.1, cas 3.

Pour la fonction $f \in F_{a,b}$ définie par

$$f(t) = \begin{cases} l_4(t) & \text{si } 0 \leq t \leq x_1 \\ l_3(t) & \text{si } x_1 \leq t \leq x_2 \\ l_1(t) & \text{si } x_2 \leq t \leq 1 \end{cases}$$

on a $\mu_1 = \frac{v_1}{x_1}$ et $\mu_2 = b$ et l'erreur pour f sur $[x_1, x_2]$ est

$$e = \frac{\delta^{1+\frac{1}{p}}}{2(1+p)^{\frac{1}{p}}} \cdot \frac{1-b}{2x_1+\delta},$$

qui est décroissante en x_1 et dont le minimum est en $1-\delta$. On obtient en substituant cette valeur dans l'expression de l'erreur du lemme 4.1

$$\begin{aligned} e &\geq \frac{1}{4(1+p)^{\frac{1}{p}}(n+1)^{\frac{1}{p}}} \cdot \frac{1-b}{2n+1} \\ &\geq \frac{1}{8(1+p)^{\frac{1}{p}}} \cdot \frac{1-b}{(n+1)^{1+\frac{1}{p}}} \\ &\geq \frac{1}{(n+1)^{1+\frac{1}{p}}} \left[\frac{(a-1)(1-b)}{12(1+p)^{\frac{1}{p}}(a-b)} \right] \end{aligned}$$

car $\frac{a-1}{a-b} \leq 1$. Ceci complète la preuve de la proposition. ∎

4.3 Algorithme adaptatif pour le problème d'approximation L^p

L'algorithme du sandwich décrit plus bas a été employé par Burkard et al. [5] et auparavant par Sonnevend [33] pour l'approximation de fonctions convexes (ou concaves) dans la norme L^∞. Cet algorithme peut aussi être utilisé pour l'approximation dans la norme L^p et la borne $\mathcal{O}(n^{-2})$ prouvée dans [5] généralisée à $1 \leq p \leq \infty$.

L'algorithme du sandwich consiste à approximer une fonction concave f à l'aide des sous- et sur- estimations U et L construites à partir de l'évaluation de f et f' en des points obtenus par une suite de bissections de l'intervalle [0,1].

Plus précisément, soit $\varepsilon > 0$ un seuil pour l'erreur a priori sur chaque sous-intervalle. Cette erreur peut être calculée à l'aide de la formule du lemme 4.1.

Algorithme du sandwich

Itération 1

Poser $x_1 = \frac{1}{2}$ et calculer $v_1 = f(x_1)$, $\mu_1 = f'(x_1)$.

Itération i

L'intervalle [0,1] est subdivisé en $i+1$ sous-intervalles $[x_j, x_{j+1}]$, $j \leq i$.

(1) Calculer l'erreur a priori e_j sur chaque sous-intervalle.

(2) Si $e_j < \varepsilon$ pour tout j, arrêter. Sinon soit j_0 tel que $e_{j_0} = \max_{j \leq i+1} e_j$.

(3) Poser $x_{i+1} = \frac{1}{2}(x_{j_0} + x_{j_0+1})$, $v_{i+1} = f(x_{i+1})$, $\mu_{i+1} = f'(x_{i+1})$.

(4) Si nécessaire, réordonner les indices de x_j, v_j, μ_j, $j \leq i+1$ de façon à ce que les x_j soient en ordre croissant. Aller à (1).

Cette procédure se termine lorsque l'erreur a priori sur chaque sous-intervalle est inférieure à ε. Si le nombre d'évaluations est fixé à l'avance, l'algorithme s'arrête lorsque ce nombre est atteint et l'erreur ε est alors bornée par la formule de la proposition 4.2 ci-dessous.

Cette méthode peut être représentée par un arbre binaire tel qu'illustré à la figure 4.7, où $n = 4$ évaluations on été effectuées pour déterminer $n+1 = 5$ sous-intervalles.

Appelons $\bar{\mathcal{A}}$ l'arbre binaire ainsi obtenu. Cet arbre comporte n sommets intérieurs correspondant chacun à une évaluation et $n+1$ feuilles correspondant chacune à un sous-intervalle. En retirant les feuilles de $\bar{\mathcal{A}}$, on obtient un nouvel arbre binaire $\mathcal{A}$ avec n feuilles, une pour chaque évaluation.

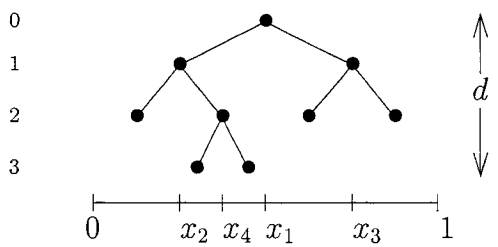


Figure 4.7 L'arbre $\bar{\mathcal{A}}$.

On définit le *niveau* d'un sommet comme étant la distance de ce sommet à la racine, la distance étant le nombre d'arêtes traversées par un chemin de la racine au sommet. La *hauteur* d'un arbre est la plus grande distance de sa racine à l'une de ses feuilles.

L'arbre $\mathcal{A}$ sera utilisé dans la preuve de la proposition 4.2 ci-dessous, mais d'abord le lemme suivant est nécessaire. C'est une généralisation du lemme 2.2 de Burkard et al. [5]. Chaque terme de la somme correspondra dans la preuve de la proposition à une borne inférieure pour la différence entre les pentes aux extrémités d'un sous-intervalle à la fin de l'algorithme, étant donné que la longueur de ce sous-intervalle est 2^{-i}. La somme de ces termes sera une borne inférieure pour la différence $a - b$, chaque feuille de niveau i représentant un sous-intervalle. Le facteur q sera tout simplement $1 + \frac{1}{p}$.

Lemme 4.2 *Soit $\mathcal{A}$ un arbre binaire de hauteur d comportant $n-1$ sommets et soit $\nu_i(\mathcal{A})$ le nombre de feuilles de $\mathcal{A}$ au niveau i. Si $q \geq 0$ alors*

$$\omega(\mathcal{A}) \equiv \sum_{i=0}^{d} 2^{qi} \nu_i(\mathcal{A}) \geq K(q) n^{q+1}$$

où

$$K(q) = \min\left\{\frac{2^q}{3^{q+1}}, \frac{1}{2^{q+1}}\right\}.$$

Preuve : D'abord, en appliquant à $\mathcal{A}$ si nécessaire les deux transformations décrites ci-dessous, on obtient un nouvel arbre binaire $\hat{\mathcal{A}}$ ayant les propriétés suivantes :

1) $\mathcal{A}$ et $\hat{\mathcal{A}}$ ont le même nombre de sommets,

2) $\omega(\mathcal{A}) = \omega(\hat{\mathcal{A}})$,

3) $\nu_i(\hat{\mathcal{A}}) = 0$ si $i \leq d-2$,

4) chaque sommet de $\hat{\mathcal{A}}$ au niveau $d-2$ possède exactement deux successeurs,

5) deux cas sont possibles :

   (A) ou bien chaque sommet au niveau $d-1$ possède au plus un successeur,

   (B) ou bien chaque sommet au niveau $d-1$ possède au moins un successeur.

Les propriétés 1) et 2) impliquent qu'il suffit de considérer les arbres du type $\hat{\mathcal{A}}$ pour prouver la borne. On supposera donc que $\mathcal{A}$ donné par l'algorithme du sandwich est de ce type. Dans ce cas par la propriété 3) la somme dans l'expression de $\omega(\mathcal{A})$ comporte seulement deux termes, ce qui simplifie les calculs.

Les transformations de $\mathcal{A}$ sont :

TRANSFORMATION 1. Pour chaque sommet y de niveau $i \leq d-2$ ayant au plus un successeur, retirer un sommet x de niveau d et en faire un successeur de y (voir figure 4.8).

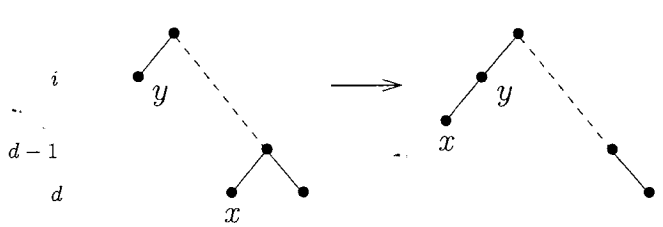


Figure 4.8 Transformation 1.

TRANSFORMATION 2. S'il existe un sommet z de niveau $d-1$ ayant deux successeurs x et y et un autre sommet u de niveau $d-1$ qui est une feuille, retirer y et en faire un sucesseur de u (voir figure 4.9).

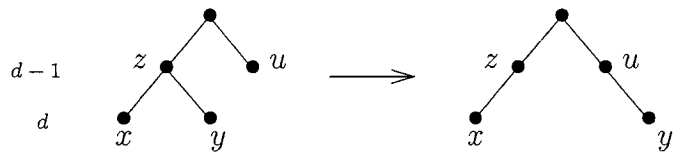


Figure 4.9 Transformation 2.

Pour la preuve que ces transformations donnent bien un arbre avec les propriétés 1) - 5), voir Burkard *et al.* [5].

Les sommets de $\mathcal{A}$ de niveau $d-1$ et moins forment un arbre binaire complet de 2^d-1 sommets. Le niveau $d-1$ contient exactement 2^{d-1} sommets et le niveau d contient $n-1-(2^d-1) = n-2^d$ sommets. Deux situations sont possibles, correspondants aux deux possibilités de la propriété 5). Dénotons par ω_A et ω_B les nombres $\omega(\mathcal{A})$ pour ces deux cas.

Cas (A) : le nombre de feuilles au niveau $d-1$ est égal au nombre de sommets sans successeurs. Comme il doit y avoir $n-2^d$ sommets ayant un successeur au niveau suivant d, on a $\nu_{d-1}(\mathcal{A}) = 2^{d-1}-(n-2^d)$, $\nu_d(\mathcal{A}) = n-2^d$ et

$$\begin{aligned}\omega_A &= 2^{q(d-1)}(2^{d-1}-(n-2^d)) + 2^{qd}(n-2^d) \\ &= 2^{q(d-1)}(2^q-1)n + 2^{(q+1)(d-1)}(3-2^{q+1}).\end{aligned}$$

Dans le cas (A) on a aussi $0 \leq n-2^d \leq 2^{d-1} \Rightarrow n \in [2^d, 2^d+2^{d-1}]$.

Cas (B) : il n'y a aucune feuille au niveau $d-1$, donc $\nu_{d-1}(\mathcal{A}) = 0$, $\nu_d(\mathcal{A}) = n - 2^d$ et

$$\omega_B = 2^{qd}(n - 2^d) = 2^{qd}n - 2^{(q+1)d}.$$

Dans le cas (B), $2^{d-1} \leq n - 2^d \leq 2^d \Rightarrow n \in [2^d + 2^{d-1}, 2^{d+1}]$.

Les nombres ω_A et ω_B sont des fonctions linéaires de n sur les intervalles déterminés ci-dessus. Pour ces fonctions $\omega_A(2^d + 2^{d-1}) = \omega_B(2^d + 2^{d-1}) = 2^{(q+1)d-1}$. De plus, $2^{qd} \geq 2^{q(d-1)}(2^q - 1)$ donc la pente de ω_B est supérieure à celle de ω_A. Ceci est illustré à la figure 4.10.

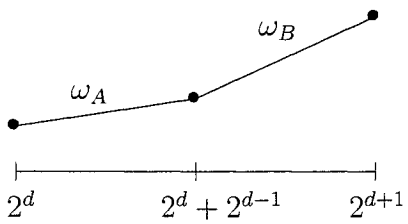


Figure 4.10 Les fonctions $\omega_A(n)$ et $\omega_B(n)$

On doit maintenant montrer que $\omega_A(n), \omega_B(n) \geq K(q)n^{q+1}$ pour tout $n \geq 1$. Puisque $q \geq 0$ le terme de droite de cette inégalité est une fonction convexe de n et il suffit de vérifier l'inégalité aux points 2^d, $2^d + 2^{d-1}$ et 2^{d+1}.

Si $n = 2^d$ alors $2^{d-1} = \dfrac{n}{2}$ et

$$\omega_A(2^d) = \left(\frac{n}{2}\right)^q (2^q - 1)n + \left(\frac{n}{2}\right)^{q+1} (3 - 2^{q+1}) = \frac{n^{q+1}}{2^{q+1}}.$$

Si $n = 2^d + 2^{d-1}$ alors $2^d = \frac{2n}{3}$ et

$$\omega_A(2^d + 2^{d-1}) = \omega_B(2^d + 2^{d-1}) = \left(\frac{2n}{3}\right)^q n - \left(\frac{2n}{3}\right)^{q+1} = n^{q+1}\left(\frac{2^q}{3^{q+1}}\right).$$

Si $n = 2^{d+1}$ alors $2^d = \frac{n}{2}$ et

$$\omega_B(2^{d+1}) = \left(\frac{n}{2}\right)^q n - \left(\frac{n}{2}\right)^{q+1} = \frac{n^{q+1}}{2^{q+1}}.$$

Chacune de ces quantités étant plus grande ou égale à $K(q)n^{q+1}$, ceci complète la preuve du lemme. ∎

Notons que $\frac{2^q}{3^{q+1}} < \frac{1}{2^{q+1}} \Leftrightarrow q < \frac{\ln(3/2)}{\ln(4/3)} \equiv q_0$. Dans le calcul de la borne sur l'erreur pour l'algorithme du sandwich, $q = 1 + \frac{1}{p} < q_0 \Leftrightarrow p > \frac{1}{1-q_0} \approx 2.4424$.

La proposition suivante est une généralisation à $p \geq 1$ du théorème 2.3 de [5].

Proposition 4.2 *Soit α^s l'algorithme du sandwich pour l'approximation sur $F_{a,b}$ dans la norme L^p, $1 \leq p \leq \infty$, avec n évaluations permises. Alors*

$$e(\alpha^s) \leq \frac{1}{(n+1)^2} \cdot \frac{a-b}{8K_p \cdot (1+p)^{\frac{1}{p}}},$$

où $K_p = K\left(1 + \frac{1}{p}\right)$ et K est défini comme au lemme 4.2.

Preuve: Si n évaluations ont été faites, chacune des $n+1$ feuilles de l'arbre $\hat{\mathcal{A}}$ obtenu à la fin de l'algorithme correspond à un sous-intervalle sur lequel l'erreur a priori est inférieure à ε. Chacun des n sommets intérieurs de $\hat{\mathcal{A}}$ correspond à un sous-intervalle qui sera subdivisé par la suite et possède exactement deux successeurs. Comme avant, soit $\mathcal{A}$ l'arbre obtenu de $\hat{\mathcal{A}}$ en lui retirant ses feuilles.

Soit Λ l'ensemble des feuilles de $\mathcal{A}$. Chaque élément $\lambda \in \Lambda$ de niveau i correspond à un sous-intervalle $[x_\lambda, \bar{x}_\lambda]$ de longueur 2^{-i} sur lequel l'erreur a priori est supérieure à ε. Si μ_λ et $\bar{\mu}_\lambda$ sont les pentes aux extrémités alors par la deuxième partie du lemme

4.1

$$\mu_\lambda - \bar{\mu}_\lambda \geq 8\varepsilon(1+p)^{\frac{1}{p}} 2^{i(1+\frac{1}{p})}.$$

Puisque les intervalles $[x_\lambda, \bar{x}_\lambda]$ sont disjoints (sauf aux extrémités) et que leur union est [0,1] on a par le lemme 4.2 avec $q = 1 + \frac{1}{p}$

$$\begin{aligned} a - b &= \sum_{\lambda \in L} \mu_\lambda - \bar{\mu}_\lambda = \sum_{i \geq 0} \nu_i(\mathcal{A}) 8\varepsilon(1+p)^{\frac{1}{p}} 2^{i(1+\frac{1}{p})} \\ &\geq 8\varepsilon(1+p)^{\frac{1}{p}} K_p (n+1)^{2+\frac{1}{p}}. \end{aligned}$$

Si $p < \infty$ alors sur chaque sous-intervalle à la fin de l'algorithme

$$\left(\int_{x_i}^{x_{i+1}} UL(x, v, \mu, t)^p \, dt \right)^{\frac{1}{p}} < \varepsilon \leq \frac{a-b}{8(1+p)^{\frac{1}{p}} K_p (n+1)^{2+\frac{1}{p}}}$$

et pour l'erreur totale

$$\begin{aligned} e(\alpha^s) &= \left(\sum_{i=0}^{n} \int_{x_i}^{x_{i+1}} UL(x, v, \mu, t)^p \, dt \right)^{\frac{1}{p}} \\ &\leq \left((n+1) \cdot \frac{(a-b)^p}{\left(8(1+p)^{\frac{1}{p}} K_p\right)^p (n+1)^{2p+1}} \right)^{\frac{1}{p}} \\ &= \frac{1}{(n+1)^2} \cdot \frac{a-b}{8K_p(1+p)^{\frac{1}{p}}}. \end{aligned}$$

Pour $p = \infty$, $\lim_{p \to \infty} (1+p)^{\frac{1}{p}} = 1$, $K_\infty = 1/2$ et sur chaque sous-intervalle

$$\max_{t \in [x_i, x_{i+1}]} UL(x, v, \mu, t) < \varepsilon \leq \frac{a-b}{8 \cdot \frac{1}{2} \cdot (n+1)^2}$$

donc

$$e(\alpha^s) = \max_{t\in[0,1]} UL(x, v, \mu, t) < \frac{a-b}{4(n+1)^2}.$$

Ceci complète la preuve de la proposition. ∎

On sait que pour l'approximation L^1 l'erreur optimale est d'ordre $\mathcal{O}(n^{-2})$. Puisque $|| f - g ||_1 \leq || f - g ||_p$ pour $p \geq 1$, l'erreur pour l'approximation L^p est au moins du même ordre. En effet, quelques soient les points optimaux pour le problème L^p, l'erreur d'approximation L^1 pour ces points est bornée inférieurement par $\frac{(a-1)(1-b)}{2(n+1)^2(a-b)}$ comme nous l'avons vu au chapitre 3. Ceci, avec la proposition (4.2), montre que l'algorithme du sandwich est d'ordre optimal.

4.4 Utilité de l'adaptation et algorithmes optimaux

Les propositions 4.1 et 4.2 montrent que l'erreur d'un algorithme passif est au mieux d'ordre $\mathcal{O}(n^{-(1+\frac{1}{p})})$ tandis qu'il existe un algorithme adaptatif avec erreur d'ordre $\mathcal{O}(n^{-2})$. On conclut donc que pour le problème d'approximation sur $F_{a,b}$ dans la norme L^p l'adaptation est utile si et seulement si $1 + \frac{1}{p} < 2$, c'est-à-dire $p > 1$.

Proposition 4.3 *Pour le problème d'approximation L^p sur $F_{a,b}$ l'adaptation est utile si et seulement si $p > 1$.*

Preuve: Ceci découle de la discussion précédent la proposition. ∎

Ceci vient compléter ce qui était déjà connu pour $p = 1$ et $p = \infty$ (voir Sonnevend [33] et Burkard et al. [5]). Il serait maintenant intéressant de calculer exactement l'erreur optimale $e^p = \inf_{\alpha\in\hat{A}} e(\alpha)$ pour l'approximation L^p. Pour $p = 1$, e^1 est connue et donnée par la formule (3.9) de la section 3.3. Pour $p > 1$ on ne connaît pas e^p. Pour le cas important $p = \infty$, Rote conclut dans [30] en posant la question de savoir

comment construire un algorithme optimal pour ce problème. Il remarque que le contexte approprié pour cette question est celui de la complexité informationnelle, mais ne propose pas de réponse.

Comme le démontre le cas $p = 1$, la programmation dynamique peut donner une formulation explicite du problème, et la solution à l'équation de récurrence est la réponse cherchée. Cependant ce cas se distingue du cas $p > 1$ sur deux points. Premièrement, puisque l'adaptation n'est pas utile au sens strict pour $p = 1$, l'ordre des points dans la formulation de programmation dynamique n'a pas d'importance, ce qui simplifie la relation de récurrence. Et deuxièmement, les calculs pour la résolution de cette récurrence s'avèrent relativement simples.

Nous donnons maintenant une formulation de programmation dynamique pour $p \geq 1$, qui généralise celle de la section 3.3. Nous proposons ensuite une simplification de la récurrence lorsque $p = \infty$.

Puisque $p \geq 1$, minimiser $|| \alpha(f) - f ||_p$ est équivalent à minimiser $(|| \alpha(f) - f ||_p)^p$. Soit $\bar{E}_p(n)(a,b)$ l'erreur optimale pour l'approximation L^p sur $F_{a,b}$ avec n évaluations et $E_p(n)(a,b) = [\bar{E}_p(n)(a,b)]^p$. En utilisant un changement de variables comme à la section 3.3, le problème pour $1 \leq p < \infty$ peut se formuler de la façon suivante

$$\begin{aligned} E_p(n)(a,b) \quad = \quad & \min_{x\in[0,1]} \max_{v,\mu\in C(x)} \min_{k+l=n-1} \left\{ xv^p E_p(k)\left(\frac{x}{v}a, \frac{x}{v}\mu\right) \right. \\ & \left. +(1-x)(1-v)^p E_p(l)\left(\frac{1-x}{1-v}\mu, \frac{1-x}{1-v}b\right) \right\}. \end{aligned} \tag{4.4}$$

Cette formulation tient compte du fait que l'ordre des points d'évaluation n'est pas fixé à l'avance et s'interprète comme suit en termes d'un jeu à deux joueurs. Le premier joueur choisit les points d'évaluation et cherche à minimiser l'erreur tandis que le deuxième choisit v et μ et cherche à la maximiser. Lorsque le premier

point x est déterminé le deuxième joueur réagit en choisissant $v, \mu \in C(x)$ de façon à maximiser l'erreur, sachant que le premier joueur pourra par la suite choisir la répartition à gauche et à droite de x des $n-1$ points restants.

Pour $p = \infty$ soit $E_\infty(n)(a,b)$ l'erreur optimale pour l'approximation L^∞ sur $F_{a,b}$ avec n évaluations. Le problème peut s'écrire

$$E_\infty(n)(a,b) = \min_{x\in[0,1]} \max_{v,\mu\in C(x)} \min_{k+l=n-1} \max \left\{ vE_\infty(k)\left(\frac{x}{v}a, \frac{x}{v}\mu\right), \right. \\ \left. (1-v)E_\infty(l)\left(\frac{1-x}{1-v}\mu, \frac{1-x}{1-v}b\right)\right\} \tag{4.5}$$

et s'interprète de la même façon que précédemment. Remarquons que pour résoudre les récurrences (4.4) et (4.5) il faut à chaque étape calculer la meilleure répartition des points restants puis les points optimaux selon cette répartition. Cette procédure donne pour chaque i un algorithme optimal à $n-i$ étapes sur le sous-ensemble de $F_{a,b}$ défini par l'information recueillie lors des i premiers calculs d'information. La solution aux récurrences produit donc, en théorie du moins, un algorithme séquentiellement optimal pour ces problèmes.

4.5 Solution pour $p = \infty$ et $n = 1$

Si $n = 1$ alors $k = l = 0$ et il est possible de calculer la solution à la récurrence (4.5) et en fait Maple peut souvent trouver une solution exacte de façon analytique. Ceci sera utilisé pour construire un algorithme optimal à chaque étape au chapitre 6, qui pourra être comparé à l'algorithme du sandwich.

Lorsque $n = 1$ les deux termes entre accolades dans (4.5) sont

$$\phi_1(\mu) = \frac{(ax - v)(v - \mu x)}{x(a - \mu)}$$

$$\phi_2(\mu) = \frac{(1 - v - (1 - x)b)(\mu(1 - x) - (1 - v))}{(1 - x)(\mu - b)}.$$

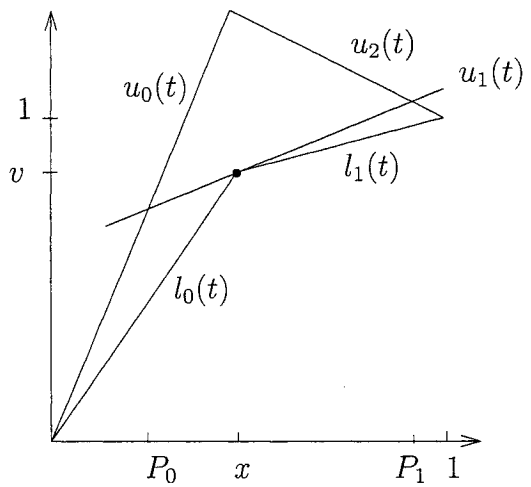


Figure 4.11 Aproximation L^∞ avec $n = 1$.

Il est facile de voir géométriquement que le maximum de ϕ_1 est en $\mu_1 = \frac{1-v}{1-x}$ et le maximum de ϕ_2 en $\mu_2 = \frac{v}{x}$.

Pour x fixé, soit

$$\psi_1(v) \equiv \phi_1(\mu_1) = \frac{(x - v)(ax - v)}{x(1 - v - a(1 - x))}$$

et

$$\psi_2(v) \equiv \phi_2(\mu_2) = \frac{(x - v)(1 - v - b(1 - x))}{(1 - x)(bx - v)},$$

que l'on cherche à maximiser sur $v \in [x, \min\{ax, bx+1-b\}]$. Soit

$$Q = \left(\frac{1-b}{a-b}\right)^2, \quad P = \frac{1-b}{a-b} \quad \text{et} \quad R = 1 - \left(\frac{a-1}{a-b}\right)^2.$$

Ces points satisfont $Q \leq P \leq R$.

On peut calculer que $\psi_1'(v) = 0 \Leftrightarrow v^{\pm} = ax - (a-1)(1 \pm \sqrt{1-x})$ et en substituant ces racines dans ψ_1'' on trouve que $\psi_1''(v^+) \geq 0$ et $\psi_1''(v^-) \leq 0$ et donc que v^+ est un minimum local et v^- un maximum. De plus il est facile de vérifier que $v^+ < x$, $v^- \geq x$, $v^- \leq ax$ et que $v^- \leq bx + 1 - b \Leftrightarrow x \leq R$. Par conséquent le maximum de ψ_1 est en

$$\begin{cases} v_1 \equiv ax - (a-1)(1-\sqrt{1-x}) & \text{si } x \leq R, \\ \\ bx + 1 - b & \text{si } x \geq R. \end{cases}$$

Ceci est illustré à la figure (4.12).

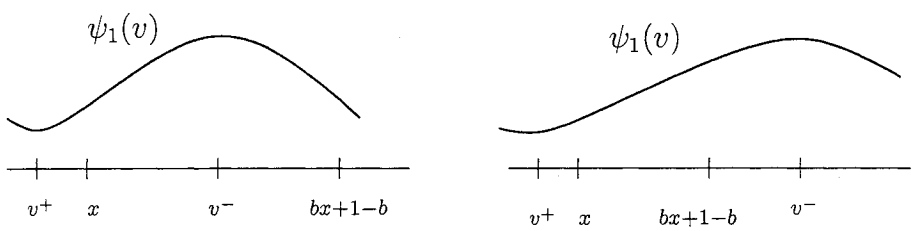


Figure 4.12 Deux cas pour le maximum de la fonction $\psi_1(v)$.

De façon semblable on peut montrer que le maximum de ψ_2 est en

$$\begin{cases} v_2 \equiv bx + (1-b)\sqrt{x} & \text{si } x \geq Q, \\ \\ ax & \text{si } x \leq Q. \end{cases}$$

La fonction à minimiser dans (4.5) est

$$\begin{aligned}\theta(x) &= \max\left\{\max_{v,\mu}\phi_1(\mu), \max_{v,\mu}\phi_2(\mu)\right\} \\ &= \max\left\{\max_{v}\psi_1(v), \max_{v}\psi_2(v)\right\}\end{aligned}$$

$$= \begin{cases} \max\{\psi_1(v_1), \psi_2(ax)\} & \text{si } x \in [0, Q] \\ \max\{\psi_1(v_1), \psi_2(v_2)\} & \text{si } x \in [Q, R] \\ \max\{\psi_1(bx+1-b), \psi_2(v_2)\} & \text{si } x \in [R, 1]. \end{cases}$$

Posons $T_1(x) = \psi_1(v_1)$, $T_2(x) = \psi_2(ax)$, $T_3(x) = \psi_2(v_2)$ et $T_4(x) = \psi_1(bx + 1 - b)$. Ces fonctions sont définies sur [0,1]. En calculant leur dérivée on trouve que T_1 et T_4 sont croissantes tandis que T_2 et T_3 sont décroissantes. De plus, $T_1(0) = 0$, $T_1(R) = T_4(R)$, $T_2(Q) = T_3(Q)$ et $T_3(1) = 0$. Ceci est illustré à la figure 4.13.

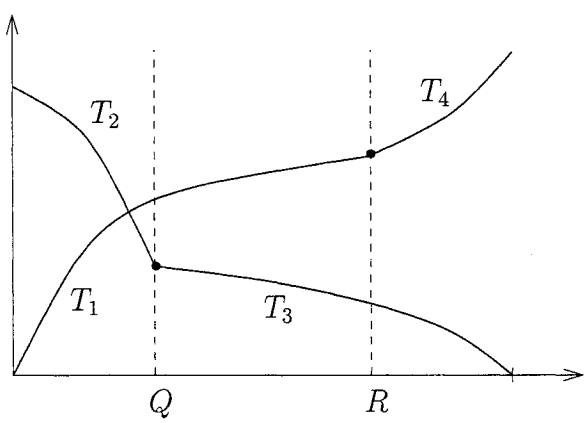


Figure 4.13 La fonction $\theta(x)$.

Enfin, soit

$$\theta_1(x) = \begin{cases} T_1(x) & x \leq R \\ T_4(x) & x \geq R \end{cases}$$

et

$$\theta_2(x) = \begin{cases} T_2(x) & x \leq Q \\ T_3(x) & x \geq Q. \end{cases}$$

Alors $\theta(x) = \max\{\theta_1(x), \theta_2(x)\}$ et puisque θ_1 est croissante et θ_2 décroissante, le minimum de θ survient au point x^* tel que $\theta_1(x^*) = \theta_2(x^*)$. Ce point dépend de a et b et peut être calculé analytiquement avec Maple lorsque ces valeurs sont données.

Proposition 4.4 *Pour le problème d'approximation L^∞ avec $n = 1$, le point x qui minimise l'erreur est le minimum de la fonction $\theta(x)$ définie ci-dessus.*

Preuve: Ceci découle de la discussion précédent la proposition. ∎

4.6 Discussion sur le cas $p = \infty$ et $n \geq 2$

Considérons maintenant le cas $n = 2$. Les deux termes entre accolades dans (4.5) sont des fonctions de x, v et μ mais on considérera d'abord que x et v sont fixés. Ces termes représentent l'erreur à gauche et à droite du premier point x. On dénotera la valeur des ces fonctions en μ par $e_j(k, \mu)$ pour $k = 0, 1$ où $j = 1$ dénote le premier terme et $j = 2$ le deuxième. Il est facile de voir que

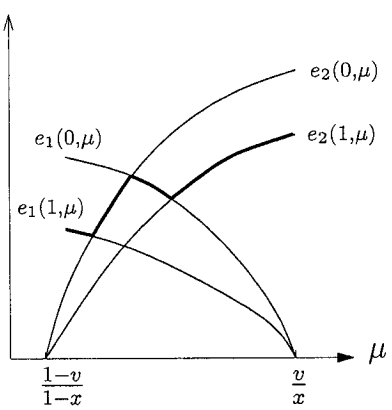
1. $e_j(0, \mu) \geq e_j(1, \mu)$ pour tout μ,
2. $e_1(0, \frac{v}{x}) = e_1(1, \frac{v}{x}) = 0$ et $e_2(0, \frac{1-v}{1-x}) = e_2(1, \frac{1-v}{1-x}) = 0$,
3. $e_1(0, \mu)$ et $e_1(1, \mu)$ sont décroissantes en μ tandis que $e_2(0, \mu)$ et $e_2(1, \mu)$ sont croissantes en μ.

Ceci est illustré à la figure 4.14. Le maximum sur μ de

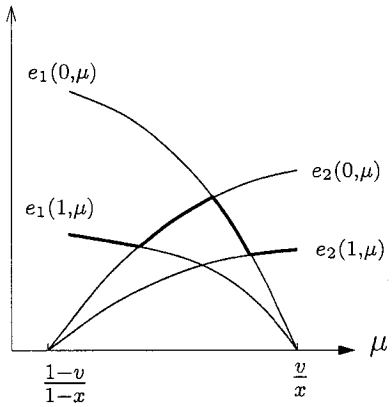
$$\min\left\{\max\{e_1(0,\mu), e_2(1,\mu)\}, \max\{e_1(1,\mu), e_2(0,\mu)\}\right\}$$

peut donc survenir lorsque

1. $\mu = \frac{1-v}{1-x}$, 2. $\mu = \frac{v}{x}$, 3. $e_1(0,\mu) = e_2(0,\mu)$.



(a) Maximum lorsque $\mu = \frac{v}{x}$



(b) Maximum lorsque $e_1(0,\mu) = e_2(0,\mu)$

Figure 4.14 Deux cas pour le maximum sur μ.

Dans le premier cas, le deuxième point est placé à gauche de x et dans le deuxième cas, à droite. On voit ici que le *principe d'égalité des erreurs* (voir Bakhvalov [1]) n'est pas satisfait pour le problème d'approximation L^∞ : l'erreur sur un des sous-intervalles peut être nulle à l'optimum tandis qu'elle ne l'est pas sur l'autre sous-intervalle. Le troisième cas correspond à une situation où le deuxième point est en fait inutile : qu'il soit placé à gauche ou à droite de x, l'erreur maximal ne sera pas réduite. Nous dénoterons par $\bar{\mu}$ la valeur de μ qui satisfait l'égalité 3.

L'ordre des points est important pour ce problème. En effet, supposons que l'ordre est fixé, par exemple de gauche à droite. Alors pour x et v fixés, le maximum sur

μ est nécessairement atteint en $\frac{1-v}{1-x}$ ou $\frac{v}{x}$, selon que la plus grande des valeurs est $e_1(0, \frac{1-v}{1-x})$ ou $e_2(1, \frac{v}{x})$. Si l'ordre n'est pas fixé à l'avance, le maximum sur μ peut possiblement être atteint en un point entre ces deux valeurs extrêmes de μ. C'est le cas si $e_1(1, \frac{1-v}{1-x})$ et $e_2(1, \frac{v}{x})$ sont inférieurs à $e_1(0, \bar{\mu})$ et alors l'erreur a (possiblement) diminué (voir le deuxième graphe sur la figure (4.14)). Ici, le deuxième point est inutile au sens où l'information supplémentaire obtenue en évaluant en ce point ne permettra pas de réduire l'erreur. Cependant, la possibilité de le placer à gauche ou à droite force μ à prendre une valeur qui donne une erreur moindre.

Dans le contexte d'un jeu à deux joueurs, la liberté qu'a le premier de choisir l'intervalle pour le deuxième point force le deuxième joueur à choisir un μ de compromis pour se prémunir contre les deux possibilités.

En conclusion, pour $n = 2$ on peut réduire le problème original (4.5) à

$$E_\infty(2)(a,b) = \min_{x\in[0,1]} \max_{v\in B(x)} \max\left\{ e_1\left(1, \frac{1-v}{1-x}\right), e_2\left(1, \frac{v}{x}\right), e_2(0,\bar{\mu})\right\}$$

où $B(x) = [x, \min\{ax, bx + 1 - b\}]$.

Pour $n > 2$, les mêmes arguments permettent de réduire le calcul de $\max_{\mu} \min_{p+q=n-1}$ dans le problème (4.5) au calcul du maximum de $n+1$ fonctions comme dans le cas $n = 2$.

La discussion précédente montre que le problème L^∞ possède un aspect combinatoire dont l'analyse permet de simplifier la récurrence (4.5). Ceci est un premier pas vers la réponse à la question posée par Rote [30]. Nous croyons qu'il serait très difficile de trouver une formule analytique pour les points d'évaluation optimaux, mais cependant même une procédure de résolution numérique efficace serait un progrès important. Dans cette optique, il serait intéressant de voir si un examen plus poussé pourrait déboucher sur de nouvelles simplifications.

CHAPITRE 5

APPROXIMATION L_h^1

Nous considérons dans cette section le problème d'approximation dans la norme L_h^1. Les données du problèmes sont les mêmes qu'au chapitre 3 sauf pour l'erreur d'approximation qui est maintenant mesurée dans la norme L^1 pondérée par une fonction de poids $h : [0,1] \rightarrow \mathbf{R}^+$. Si $f, g \in F_{a,b}$ alors

$$|| f - g ||_{1,h} = \int_0^1 | f(t) - g(t) | h(t)\, dt.$$

Les résultats de la section 3.1 concernant l'opérateur d'information et l'opération terminale sont encore valides. En particulier, ϕ^* est centrale et donc optimale.

Comme avant, dénotons $UL = \frac{1}{2}(U - L)$. Pour n donné, $x, f(x)$ et $f'(x)$ dénotent les vecteurs $(x_1, \ldots, x_n)$, $(f(x_1), \ldots, f(x_n))$ et $(f'(x_1), \ldots, f'(x_n))$ respectivement.

5.1 Utilité de l'adaptation

Nous montrons dans cette section que pour l'approximation L_h^1 l'adaptation n'est pas utile au sens large si la fonction de poids h est strictement positive. Intuitivement, ceci découle du fait que les problèmes L_h^1 et L^1 sont semblables, le second étant un cas particulier du premier. L'introduction d'une fonction de poids h connue n'augmente pas l'incertitude sur la fonction à approximer et puisque que l'adaptation n'est pas utile pour l'approximation L^1, elle ne l'est pas non plus pour L_h^1. Ceci est démontré rigoureusement ci-dessous.

Proposition 5.1 *1. L'erreur optimale pour le problème d'approximation L_h^1 avec $h > 0$ est d'ordre $\mathcal{O}(n^{-2})$.*

2. Pour ce problème l'adaptation n'est pas utile au sens large.

Preuve: D'abord, dénotons $E_{1,h}(n)(a,b)$ et $E_1(n)(a,b)$ les erreurs optimales pour l'approximation L_h^1 et L^1 avec n points sur $F_{a,b}$, respectivement. Soit x_f^* les points d'évaluation optimaux pour la fonction $f \in F_{a,b}$ donnés par un algorithme optimal pour le problème L_h^1. L'indice f reflète le fait que ces points peuvent dépendre de f dans le cas d'un algorithme adaptatif. Si $\underline{h}$ et $\overline{h}$ sont les minimum et maximum essentiels de h sur [0,1] alors pour tout $f \in F_{a,b}$ on a

$$\underline{h} \cdot \int_0^1 UL(x_f^*, f, t)\, dt \leq \int_0^1 UL(x_f^*, f, t)h(t)\, dt$$

et en prenant le supremum sur f de chaque côté de cette inégalité on obtient

$$\underline{h}E_1(n)(a,b) \leq \underline{h} \cdot \sup_{f \in F_{a,b}} \int_0^1 UL(x_f^*, f, t)\, dt = E_{1,h}(n)(a,b).$$

Par la formule (3.9) de la section 3.3 pour $E_1(n)(a,b)$ on obtient

$$\underline{h}\frac{(a-1)(1-b)}{2(n+1)^2(a-b)} \leq E_{1,h}(n)(a,b). \tag{5.1}$$

L'erreur $E_{1,h}(n)(a,b)$ est donc d'ordre au moins $\mathcal{O}(n^{-2})$.

D'autre part, soit x^* les points d'évaluation optimaux pour un algorithme (passif) optimal pour l'approximation L^1. Alors pour tout $f \in F_{a,b}$

$$\int_0^1 UL(x^*, f, t)h(t)\, dt \leq \overline{h} \int_0^1 UL(x^*, f, t)\, dt$$

et en prenant le supremum sur f,

$$E_{1,h}(n)(a,b) \leq \overline{h} \sup_{f \in F_{a,b}} \int_0^1 UL(x^*, f, t)h(t)\, dt = \overline{h} E_1(n)(a,b)$$

et on a

$$E_{1,h}(n)(a,b) \leq \overline{h} \frac{(a-1)(1-b)}{2(n+1)^2(a-b)}, \tag{5.2}$$

ce qui montre qu'il existe un algorithme passif avec erreur d'ordre $\mathcal{O}(n^{-2})$ pour le problème L_h^1. En combinant les inégalités (5.1) et (5.2) on obtient 1 et 2 de la proposition. ∎

5.2 Simplification du problème

Grâce à la proposition 5.1 le problème d'approximation L_h^1 sur $F_{a,b}$ s'énonce

$$E_{1,h}(n)(a,b) = \min_{x \in [0,1]^n} \sup_{f \in F_{a,b}} \int_0^1 UL(x, f, t)h(t)\, dt.$$

En posant $v = f(x)$ et $\mu = f'(x)$ le problème devient

$$E_{1,h}(n)(a,b) = \min_{x \in [0,1]^n} \max_{v,\mu \in C(x)} \int_0^1 UL(x, v, \mu, t)h(t)\, dt. \tag{5.3}$$

L'ensemble $C(x)$ décrit les contraintes que doivent satisfaire v et μ du fait que la fonction à approximer f est concave, tel que défini à la section 3.2.

Ce problème min-max est difficile à résoudre pour deux raisons. D'abord, la fonction de x à minimiser peut être très compliquée, dépendant de h. Ensuite, le domaine des variables v et μ dépend des variables x ce qui fait que les conditions d'optimalité habituellement employées pour ce type de problème ne sont pas applicables ici. Notons que bien que $C(x)$ soit un polyèdre convexe pour chaque x, le domaine

$\{(x, v, \mu) \mid x \in [0,1]^n, (v, \mu) \in C(x)\}$ n'est ni un polyèdre, ni convexe.

Nous allons maintenant chercher à simplifier le problème (5.3). Dans la suite nous allons toujours supposer que le nombre n de points d'évaluation est fixé. Les points $x_1, \ldots, x_n$ à déterminer subdivisent l'intervalle [0,1] en n sous-intervalles $[x_{i-1}, x_{i+1}]$ et sur chacun on peut définir un problème d'approximation L_h^1 avec $n = 1$ en fixant $v_{i-1}, \mu_{i-1}, v_{i+1}, \mu_{i+1}$, le point à choisir étant x_i. Ce problème se ramène à la situation standard sur [0,1] par un changement de variables. Cette décomposition du problème sur [0,1] en sous-problèmes sera cruciale pour la suite. Ceci est illustré à la figure 5.1, où $u_i(t) = \mu_i(t - x_i) + v_i$ et $l_i(t) = \left(\frac{v_{i+1}-v_i}{x_{i+1}-x_i}\right)(t - x_i) + v_i$.

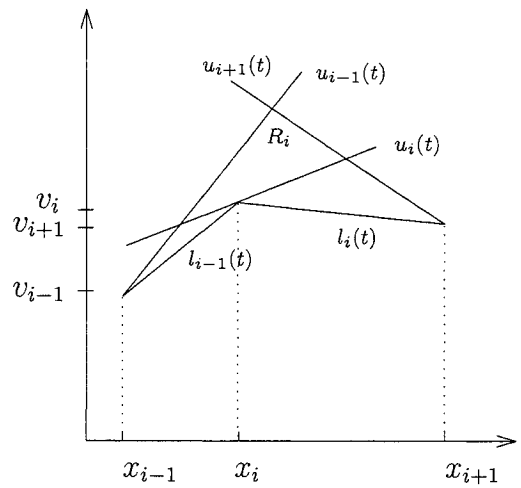


Figure 5.1 Le problème sur $[x_{i-1}, x_{i+1}]$.

Lemme 5.1 *Si $x = (x_1, \ldots, x_n)$ est un minimum pour le problème (5.3) alors pour chaque i, x_i est le minimum pour le problème avec $n = 1$ défini sur $[x_{i-1}, x_{i+1}]$ par $v_{i-1}, v_{i+1}, \mu_{i-1}, \mu_{i+1}$.*

Preuve: Si x_i n'est pas optimal sur $[x_{i-1}, x_{i+1}]$ alors on peut trouver $\bar{x}_i$ avec $\bar{v}_i$ et $\bar{\mu}_i$ correspondants qui réduirait l'erreur sur ce sous-intervalle. Clairement, changer

x_i, v_i, μ_i en gardant fixées les valeurs aux extrémités du sous-intervalle ne change pas l'erreur sur le reste de [0,1]. Par conséquent le nouveau $\bar{x}_i$ diminue l'erreur totale, contredisant le fait que les x_i sont optimaux. ∎

Soit R_i l'intersection des deux tangentes $u_{i-1}(t)$ et $u_{i+1}(t)$ (voir la figure 5.1). Si $(x_i, v_i) = R_i$ alors v_i n'est pas maximal car l'erreur sur l'intervalle est alors nulle. Dans la suite on supposera donc que $(x_i, v_i) \neq R_i$. De même on supposera toujours que $x_{i-1} < x_i < x_{i+1}$ car sinon le point x_i est inutile et n'est donc pas optimal. De plus, si $a = 1$ ou $b = 1$ alors $F_{a,b}$ ne contient que la fonction identité et le problème d'approximation ne se pose pas. On suppose donc que $a > 1$ et $b < 1$.

Nous allons maintenant examiner le problème sur un sous-intervalle lorsqu'un seul point d'évaluation est à déterminer. Pour simplifier, on considérera la situation standard sur [0,1], quitte à se ramener au cas général par un changement de variables. Soit

$$G(x, v, \mu) = \int_0^1 UL(x, v, \mu, t)h(t)\, dt$$

l'erreur sur [0,1] en fonction de x, v, μ (qui sont des scalaires puisque $n = 1$). Comme avant, dénotons par P l'abscisse de l'intersection des tangentes aux extrémités $u_0(t) = at$ et $u_2(t) = bt + 1 - b$ et soit $R = (P, u_0(P))$. Définissons aussi les abscisses P_0 et P_1 des points d'intersection de ces tangentes avec la tangente en x, $u_1(t) = \mu(t - x) + v$, tel qu'illustré à la figure 5.1. Explicitement,

$$P_0 = \frac{v - \mu x}{a - \mu}, \quad P = \frac{1 - b}{a - b}, \quad P_1 = \frac{\mu x - b + 1 - v}{\mu - b}.$$

Avec cette notation,

$$\begin{aligned} G(x,v,\mu) &= \int_0^{P_0} \left(a - \frac{v}{x}\right) th(t)\,dt + \int_{P_0}^{x} \left[\left(\mu - \frac{v}{x}\right)(t-x) + v\right] h(t)\,dt \\ &\quad + \int_x^{P_1} \left(\mu - \frac{1-v}{1-x}\right)(t-x)h(t)\,dt \qquad (5.4) \\ &\quad + \int_{P_1}^{1} \left[\left(b - \frac{1-v}{1-x}\right)(t-x) - b(1-x) + 1 - v\right] h(t)\,dt. \end{aligned}$$

Cette fonction est continue pour tout x, v, μ sauf en $\mu = a$ et en $\mu = b$. En fait G est aussi définie en (x, ax, a) et en $(x, bx + 1 - b, b)$ pour tout x. Pour le premier de ces points les deux derniers termes de (5.4) sont bien définis lorsque $\mu = a$. Quant aux deux premiers, ils représentent l'erreur sur l'intervalle $[0, x]$, qui est nulle si $v = ax$ et $\mu = a$. $G(x, ax, a)$ est donc égal aux deux derniers termes évalués en (x, ax, a). De la même façon on montre que G est définie en $(x, bx + 1 - b, b)$.

Les dérivées de G sont

$$\begin{aligned} G_x(x,v,\mu) &= \frac{m_0}{x}\int_0^x th(t)\,dt + \frac{m_1}{1-x}\int_x^1 (1-t)h(t)\,dt - \mu\int_{P_0}^{P_1} h(t)\,dt, \\ G_v(x,v,\mu) &= \int_{P_0}^{P_1} h(t)\,dt - \frac{1}{x}\int_0^x th(t)\,dt - \frac{1}{1-x}\int_x^1 (1-t)h(t)\,dt, \qquad (5.5) \\ G_\mu(x,v,\mu) &= \int_{P_1}^{P_0} (t-x)h(t)\,dt. \end{aligned}$$

où $m_0 = \dfrac{v}{x}$ et $m_1 = \dfrac{1-v}{1-x}$. Ces dérivées sont définies pour tous x, v et μ sauf si $\mu = a$ ou $\mu = b$. Pour x fixé, même si le gradient $\nabla_{v,\mu} G$ par rapport à v et μ n'est pas défini en (ax, a) et $(bx + 1 - b, b)$, la dérivée directionnelle de G existe en ces points

pour toute direction $\vec{d} = (d_1, d_2)$ avec $d_2 \neq 0$. Pour le point (ax, a) on a

$$\begin{aligned} G_{\vec{d}}(x, ax, a) &= \lim_{s \to 0} \frac{G(x, ax + d_1 s, a + d_2 s) - G(x, ax, a)}{s \,||\, \vec{d} \,||} \\ &= -\frac{d_1}{x} \int_0^{\frac{d_2 x - d_1}{d_2}} t h(t)\, dt - \frac{d_2 x - d_1}{x} \int_{\frac{d_2 x - d_1}{d_2}}^{x} (x - t) h(t)\, dt \\ &\quad - \frac{d_2(1 - x) + d_1}{1 - x} \int_x^P (x - t) h(t)\, dt - \frac{d_1}{1 - x} \int_P^1 (1 - t) h(t)\, dt. \end{aligned} \tag{5.6}$$

Cette expression a été obtenue en appliquant la règle de l'Hospital et en utilisant Maple pour les calculs. Une autre propriété de G est donnée par la proposition suivante :

Proposition 5.2 *Pour chaque $x \neq P$ la fonction G est strictement concave en v et en μ sur $C(x)$. Pour $x = P$, G est strictement concave sur $C(P)$ sauf possiblement lorsque $v = aP$.*

Preuve: D'abord en dérivant à nouveau la dérivée en v donnée en (5.5) on trouve

$$G_{vv}(x, v, \mu) = -\left(\frac{h(P_1)}{\mu - b} + \frac{h(P_0)}{a - \mu} \right) < 0$$

pour $h > 0$ et $b < \mu < a$, ce qui est vérifié pour $v, \mu \in C(x)$.

Ensuite, le déterminant du hessien de G par rapport à v, μ est

$$\det(\mathrm{Hess}_{v,\mu}(G)) = h(P_1) h(P_0)(P_1 - P_0)(P_{1v} P_{0\mu} - P_{1\mu} P_{0v}).$$

Or $h > 0$ et $P_1 - P_0 > 0$ si $(x, v) \neq R$, donc ce déterminant est strictement positif

si et seulement si

$$P_{1v}P_{0\mu} - P_{1\mu}P_{0v} > 0. \tag{5.7}$$

Un calcul montre que ceci est vérifié si et seulement si $aP > \mu(P - x) + v$, ce qui est toujours vrai si $(x, v) \neq R = (P, aP)$. Dans ce cas le hessien de G est donc défini négatif, ce qui montre que G est strictement concave.

Si $(x, v) = R$, ce qui est possible seulement si $x = P$, alors l'inégalité (5.7) n'est pas stricte et le hessien est semi-défini négatif. ∎

La remarque qui suit la proposition 5.1 montre qu'en fait le cas $(x, v) = R$ ne sera jamais considéré et par conséquent on supposera toujours que G est strictement concave.

La proposition suivante montre que les contraintes $C(x)$ ne jouent pas de rôle lors de la maximisation sur v, μ pour les x qui sont candidats pour le minimum.

Proposition 5.3 *Si x est un minimum pour le problème (5.3) alors le maximum sur v, μ est atteint en un point où $G_v(x, v, \mu) = G_\mu(x, v, \mu) = 0$.*

Preuve: Montrons que pour x_0 fixé si le maximum sur v, μ est atteint en un point de la frontière de $C(x_0)$ qui n'est pas un point critique alors x_0 n'est pas un minimum pour (5.3).

Considérons d'abord le cas où $x_0 \leq P$. On a alors $\min\{ax_0, bx_0 + 1 - b\} = ax_0$ et

$$C(x_0) : \begin{cases} x_0 \leq v \leq ax_0 \\ \\ \dfrac{1 - v}{1 - x_0} \leq \mu \leq \dfrac{v}{x_0}. \end{cases}$$

Le domaine $C(x_0)$ est illustré à la figure 5.2. Sur cette même figure est aussi illustré

le domaine $C(x_\epsilon)$ pour un petit $\epsilon > 0$, où $x_\epsilon = x_0 + \epsilon$.

On doit examiner six cas pour (v, μ) sur la frontière de $C(x_0)$, correspondant aux trois segments la composant et aux extrémités de ces segments.

Cas 1. Si $v = x_0$ alors on doit avoir $\mu = 1$ et l'erreur est réduite à 0, qui n'est pas le maximum.

Cas 2. Si $v = ax_0$ alors il est clair géométriquement que $\mu = a$ est maximal et cette situation sera examinée plus loin.

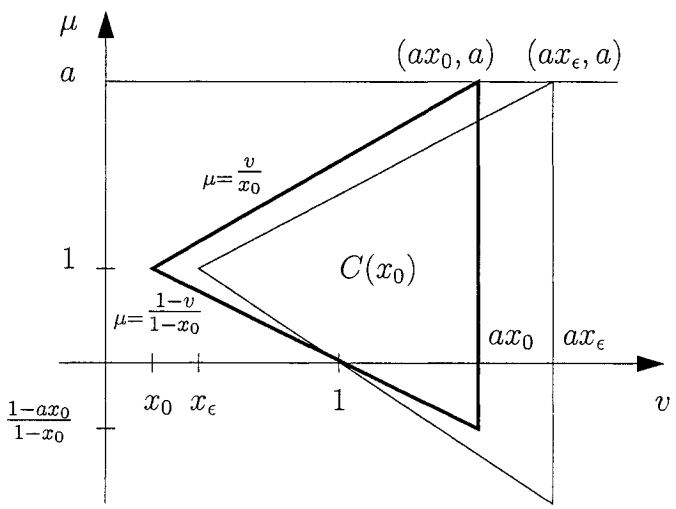


Figure 5.2 Le domaine $C(x_0)$.

Cas 3. Soit x_0 fixé. Si le maximum est atteint lorsque $\mu = \frac{v}{x_0}$ (et n'est pas un point critique) alors au point optimal on a

$$\begin{cases} \nabla_{v,\mu} G(x_0, v, \mu) &= \lambda \nabla_{v,\mu} l_1(x_0, v, \mu) \\ l_1(x_0, v, \mu) &= 0 \\ \lambda &> 0, \end{cases} \tag{5.8}$$

où $l_1(x_0, v, \mu) = \mu - \dfrac{v}{x_0}$ et $\nabla_{v,\mu} l_1(x_0, v, \mu) = (-\dfrac{1}{x_0}, 1)$. Ici $\nabla_{v,\mu}$ dénote le gradient par rapport aux variables (v, μ).

Le point v_0 satisfaisant les conditions (5.8) correspond à l'intersection des deux courbes $\lambda = g_1(v) \equiv -x_0 G_v(x_0, v, \frac{v}{x_0})$ et $\lambda = g_2(v) \equiv G_\mu(x_0, v, \frac{v}{x_0})$, tel qu'illustré à la figure 5.3. Si on perturbe les équations (5.8) en ajoutant un petit $\epsilon > 0$ à x_0, ce système aura encore une solution avec $v \in]x_\epsilon, ax_\epsilon[$ et $\lambda > 0$ à condition que l'intersection des courbes de la figure 5.3 soit transversale. Nous allons montrer que c'est le cas en montrant que ces courbes ne sont jamais tangentes.

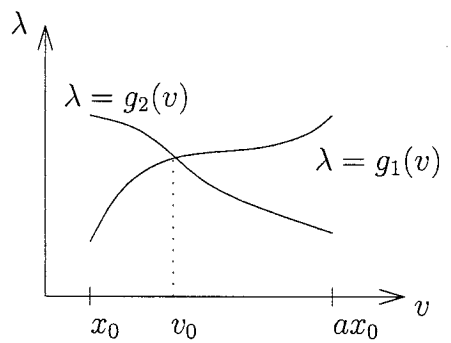


Figure 5.3 Solution aux équations (5.8)

Les dérivées de G en $(v, \frac{v}{x_0})$ sont

$$G_v(x_0, v, \tfrac{v}{x_0}) = \int_0^{Q_0} h(t)\,dt - \frac{1}{x_0}\int_0^{x_0} th(t)\,dt - \frac{1}{1-x_0}\int_{x_0}^{1} (1-t)h(t)\,dt,$$

$$G_\mu(x_0, v, \tfrac{v}{x_0}) = \int_0^{Q_0} (t - x_0)h(t)\,dt,$$

où $Q_0 = \dfrac{(1-b)x_0}{v-bx_0} > 0$. Si $\lambda = -x_0 G_v(x_0, v, \frac{v}{x_0})$ alors

$$\frac{d\lambda}{dv} = -x_0 h(Q_0)\frac{dQ_0}{dv} = x_0 h(Q_0)\frac{Q_0}{v-bx_0} \tag{5.9}$$

et si $\lambda = G_\mu(x_0, v, \frac{v}{x_0})$ alors

$$\frac{d\lambda}{dv} = h(Q_0)(Q_0 - x_0)\frac{dQ_0}{dv} = -h(Q_0)(Q_0 - x_0)\frac{Q_0}{v-bx_0}. \tag{5.10}$$

Par hypothèse $h(Q_0) > 0$ et $Q_0 > 0$ puisque $(1-b)x_0 > 0$ et $v - bx_0 > 0$. Les dérivées (5.9) et (5.10) sont donc égales si et seulement si $x_0 = x_0 - Q_0 \Leftrightarrow Q_0 = 0$, ce qui est impossible.

De ceci on conclut que pour $\epsilon > 0$ assez petit le système (5.8) perturbé possède une solution v_ϵ avec $v_\epsilon \in]x_\epsilon, ax_\epsilon[$ et $\lambda > 0$. La fonction G étant concave en (v, μ) sur $C(x_\epsilon)$, ces conditions impliquent que ce point est un maximum. On a donc montré que si $x_0 \leq P$ et que le maximum en (v, μ) est atteint lorsque $\mu = \frac{v}{x_0}$ alors il en est de même pour x_ϵ.

Soit $(v_0, \frac{v_0}{x_0})$ le maximum correspondant à x_0. Alors ce point est le maximum sur v de la fonction $G(x_0, v, \frac{v}{x_0})$ sur $[x_0, ax_0]$. Le cas $v = x_0$ a déjà été examiné et le cas $v = ax_0$ le sera plus loin. Supposons donc que le maximum est atteint en un point critique v_0, c'est-à-dire que

$$0 = \frac{d}{dv} G\left(x_0, v, \tfrac{v}{x_0}\right)\Big|_{v=v_0} = \frac{1}{x_0(1-x_0)}\int_{x_0}^{Q_0}(t-x_0)h(t)\,dt - \frac{1}{1-x_0}\int_{Q_0}^{1}(1-t)\,dt$$

$$\Leftrightarrow \quad \int_{x_0}^{Q_0}(t-x_0)h(t)\,dt = x_0\int_{Q_0}^{1}(1-t)h(t)\,dt. \tag{5.11}$$

Substituant le membre de gauche dans l'expression de $G(x_0, v, \frac{v}{x_0})$ on trouve après

simplifications que pour v_0 maximal satisfaisant (5.11)

$$G\left(x_0, v_0, \tfrac{v_0}{x_0}\right) = (1-b)\int_{Q_0}^{1}(1-t)h(t)\,dt.$$

De même pour $\epsilon > 0$ assez petit, pour le maximum $(v_\epsilon, \frac{v_\epsilon}{x_\epsilon})$ correspondant à x_ϵ

$$G\left(x_\epsilon, v_\epsilon, \tfrac{v_\epsilon}{x_\epsilon}\right) = (1-b)\int_{Q_\epsilon}^{1}(1-t)h(t)\,dt,$$

où $Q_\epsilon = \dfrac{(1-b)x_\epsilon}{v_\epsilon - bx_\epsilon}$. Clairemement, $G(x_\epsilon, v_\epsilon, \frac{v_\epsilon}{x_\epsilon}) < G(x_0, v_0, \frac{v_0}{x_0})$ si $Q_\epsilon > Q_0$. Ceci est vérifié si

$$\frac{x_\epsilon}{v_\epsilon - bx_\epsilon} > \frac{x_0}{v_0 - bx_0} \Leftrightarrow v_\epsilon < \frac{v_0}{x_0}x_\epsilon. \tag{5.12}$$

Géométriquement, ceci signifie que (x_ϵ, v_ϵ) est sous la droite de pente $\frac{v_0}{x_0}$ et d'origine (0,0) (voir figure 5.4).

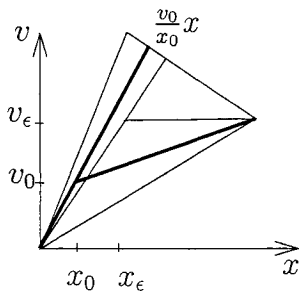


Figure 5.4 Proposition 5.3, cas 3.

La fonction $G(x_\epsilon, v, \frac{v}{x_\epsilon})$ étant concave, pour démontrer la deuxième inégalité de (5.12), il suffit de montrer

$$\frac{d}{dv}G(x_\epsilon, v, \tfrac{v}{x_\epsilon})\Big|_{v=\frac{v_0}{x_0}x_\epsilon} < 0. \tag{5.13}$$

Puisque v_0 est maximum pour x_0 il satisfait la condition (5.11). Dénotons par A le membre de gauche de cette inégalité et par B le membre de droite. Un calcul montre

que (5.13) est satisfaite si

$$C \equiv \int_{x_\epsilon}^{Q_0} (t - x_\epsilon) h(t)\, dt < x_\epsilon \int_{Q_0}^{1} (1-t) h(t)\, dt \equiv D. \tag{5.14}$$

Or pour $\epsilon > 0$ on a clairement $A > C$ et $D > B$. Donc $C < A = B < D$ et (5.14) est bien satisfaite.

En conclusion, on a montré que l'erreur en x_ϵ est inférieure à celle en x_0. Le point x_0 ne peut donc pas être un minimum pour le problème (5.3).

Cas 4. Si le maximum sur (v, μ) de G est atteint lorsque $\mu = \frac{1-v}{1-x_0}$, les arguments sont semblables à ceux du cas précédent. Soit $l_2(v, \mu) = \frac{1-v}{1-x_0} - \mu$ et $\nabla_{v,\mu} l_2 = \left(-\frac{1}{1-x_0}, -1\right)$. Les conditions d'optimalité sont

$$\begin{cases} G_v(x_0, v, \frac{1-v}{1-x_0}) &= \lambda - \dfrac{\lambda}{1-x_0} \\ G_\mu(x_0, v, \frac{1-v}{1-x_0}) &= -\lambda \\ \lambda > 0. \end{cases} \tag{5.15}$$

Comme auparavant, on peut montrer que ces conditions sont encore satisfaites si on perturbe x_0 en lui retranchant un petit $\epsilon > 0$. On a

$$G\left(x_0, v, \tfrac{1-v}{1-x_0}\right) = \frac{a x_0 - v}{x} \int_0^{R_0} t h(t)\, dt + \frac{v - x_0}{x_0(1-x_0)} \int_{R_0}^{x_0} (x_0 - t)\, dt$$

où $R_0 = \dfrac{v - x_0}{a(1-x_0) + v - 1}$. La condition $\dfrac{d}{dv} G\left(x_0, v, \frac{1-v}{1-x_0}\right) = 0$ se simplifie pour donner

$$(1-x_0) \int_0^{R_0} t h(t)\, dt = \int_{R_0}^{x_0} (x_0 - t) h(t)\, dt. \tag{5.16}$$

En substituant le membre de droite dans l'expression de $G(x_0, v, \frac{1-v}{1-x_0})$ on trouve

qu'au maximum v_0

$$G\left(x_0, v_0, \tfrac{1-v_0}{1-x_0}\right) = (a-1)\int_0^{R_0} th(t)\,dt.$$

L'erreur $G\left(x_\epsilon, v_\epsilon, \tfrac{1-v_\epsilon}{1-x_\epsilon}\right)$ en $x_\epsilon = x_0 - \epsilon$ est inférieure à celle en x_0 si $R_0 > R_\epsilon$, où R_ϵ est l'analogue de R_0 pour x_ϵ. C'est le le cas si

$$v_\epsilon < v_0 - \epsilon\left(\frac{1-v_0}{1-x_0}\right).$$

(Voir la figure 5.5.(

Or

$$\frac{d}{dv}G\left(x_\epsilon, v, \tfrac{1-v}{1-x_\epsilon}\right)\Big|_{v=v_0-\epsilon\left(\frac{1-v_0}{1-x_0}\right)} < 0$$

si

$$(1-x_\epsilon)\int_0^{R_0} th(t)\,dt > \int_{R_0}^{x_\epsilon}(x_\epsilon - t)h(t)\,dt. \tag{5.17}$$

Le membre de gauche de (5.17) est supérieur au membre de gauche de (5.16) et le membre de droite de (5.17) est inférieur à celui de (5.16). L'inégalité (5.17) est donc satisfaite.

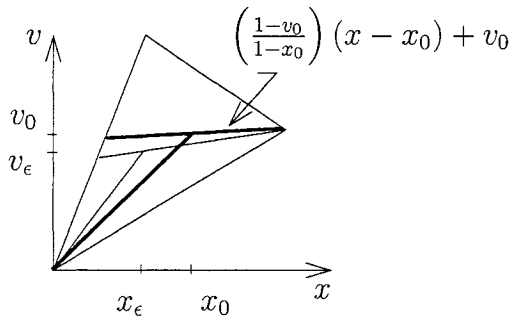


Figure 5.5 Proposition 5.3, cas 4.

On conclut que l'erreur en x_ϵ est inférieure à celle en x_0 et ce point ne peut être un minimum pour le problème (5.3).

Cas 5. Examinons maintenant le cas où le maximum sur v, μ de G est atteint en (ax_0, a). Soit $l_1(v, \mu) = \mu - \frac{v}{x_0}$ comme avant et $l_3(v, \mu) = v - ax_0$. Soit aussi $\vec{d_1} = (x_0, 1)$ et $\vec{d_2} = (0, 1)$ les vecteurs illustrés sur la figure 5.6. À l'aide de la formule (5.6) on peut calculer les dérivées de G dans les directions $\vec{d_1}$ et $\vec{d_2}$ en (ax_0, a) :

$$G_{\vec{d_1}}(x_0, ax_0, a) = \frac{1}{(1-x_0)\sqrt{1+x_0^2}} \left[\int_{x_0}^{P} (t - x_0)h(t)\, dt - x_0 \int_{P}^{1} (1-t)h(t)\, dt \right],$$

$$G_{\vec{d_2}}(x_0, ax_0, a) = \int_{x_0}^{P} (t - x_0)h(t)\, dt > 0.$$

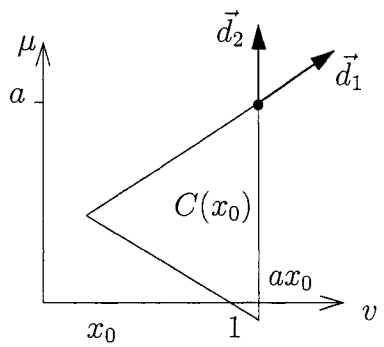


Figure 5.6 Proposition 5.3, cas 5 : les vecteurs $\vec{d_1}$ et $\vec{d_2}$.

Puisque le maximum est en (ax_0, a), G est croissante dans la direction $\vec{d_1}$ en ce point, donc

$$\int_{x_0}^{P} (t - x_0)h(t)\, dt \geq x_0 \int_{P}^{1} (1-t)h(t)\, dt. \qquad (5.18)$$

Le membre de gauche de cette inégalité est décroissant en x_0 et le membre de droite est croissant. Si l'inégalité est stricte, elle est toujours valide lorsque x_0 est augmenté à x_ϵ, à condition que ϵ soit assez petit. Ceci implique que $G_{\vec{d_1}}(x_\epsilon, ax_\epsilon, a) > 0$, ce qui

implique à son tour, G étant concave, que le maximum sur $C(x_\epsilon)$ est en (ax_ϵ, a). Dans ce cas il est clair que l'erreur en x_ϵ est inférieure à celle en x_0 (voir figure 5.7) et donc que ce point n'est pas un minimum pour le problème (5.3).

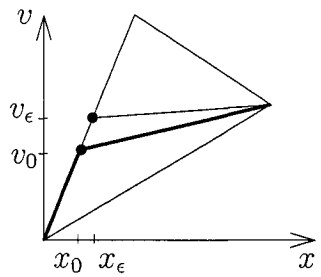


Figure 5.7 Proposition 5.3, cas 5 : l'erreur en x_0 et x_ϵ.

Enfin, si $G_{\vec{d}_1}(x_0, ax_0, a) = 0$, montrons que pour x_ϵ avec $\epsilon > 0$ assez petit le maximum sur v, μ est atteint lorsque $\mu = \frac{v}{x_\epsilon}$. On se retrouve alors dans la situation du cas 3 et on a vu que ceci implique que x_0 n'est pas un minimum. Pour ce faire on doit montrer que les équations (5.8) sont satisfaites en x_ϵ. Ces équations sont équivalentes à

$$-x_\epsilon G_v\left(x_\epsilon, v, \tfrac{v}{x_\epsilon}\right) = G_\mu\left(x_\epsilon, v, \tfrac{v}{x_\epsilon}\right) > 0$$

et un calcul permet de voir que l'égalité ci-dessus est équivalente à

$$x_\epsilon \int_{Q_\epsilon}^{1} (1-t)h(t)\,dt = \int_{x_\epsilon}^{Q_\epsilon} (t-x_\epsilon)h(t)\,dt. \tag{5.19}$$

Si $G_{\vec{d}_1}(x_0, ax_0, a) = 0$ alors (5.18) est satisfaite à égalité et pour tout $\epsilon > 0$ on a

$$x_\epsilon \int_{P}^{1} (1-t)h(t)\,dt > \int_{x_\epsilon}^{P} (t-x_\epsilon)h(t)\,dt. \tag{5.20}$$

Rappelons que $Q_\epsilon = \frac{(1-b)x_\epsilon}{v-bx_\epsilon}$ et notons que $Q_\epsilon = 1$ si $v = x_\epsilon$ et $Q_\epsilon = P$ si $v = ax_\epsilon$. De plus, si v décroît à partir de ax_ϵ alors Q_ϵ croît à partir de P. Donc (5.20) implique

qu'il existe $\bar{v}$ et $\bar{Q}_\epsilon$ correspondant tel que

$$\int_{x_\epsilon}^{\bar{Q}_\epsilon} (t - x_\epsilon)h(t)\,dt = x_\epsilon \int_{\bar{Q}_\epsilon}^{1} (1-t)h(t)\,dt.$$

Si ϵ est assez petit alors on peut trouver $\bar{v}$ proche de ax_ϵ de sorte que $G_\mu(x_\epsilon, \bar{v}, \frac{\bar{v}}{x_\epsilon}) > 0$ car cette fonction de v est continue et strictement positive en ax_ϵ. Ceci montre que (5.19) est vérifiée et que les conditions (5.8) sont staisfaites en $(x_\epsilon, \bar{v}, \frac{\bar{v}}{x_\epsilon})$ et il s'ensuit que le maximum de G sur v, μ est atteint en ce point.

Cas 6. Finalement, le maximum sur v, μ ne peut survenir en $(ax_0, \frac{1-ax_0}{1-x_0})$ car l'erreur est alors réduite à zéro.

Ceci achève la preuve de la proposition dans le cas où $x_0 \leq P$. Si $x_0 > P$, les arguments sont semblables et ne seront pas présentés ici. ∎

Les propositions 5.2 et 5.3 impliquent que pour x minimum la condition pour le maximum sur v, μ est simplement $\nabla_{v,\mu} G(x, v, \mu) = 0$. La proposition suivante montre qu'on peut encore simplifier le problème.

Proposition 5.4 *Si (x^*, v^*, μ^*) est une solution du problème (5.3) pour $n = 1$ alors* $\nabla G(x^*, v^*, \mu^*) = 0$.

Ici ∇ dénote le gradient par rapport à toutes les variables x, v et μ.

Preuve: Soit x^* un minimum et v^*, μ^* correspondants. Définissons la fonction S : $\mathbf{R} \times \mathbf{R}^2 \to \mathbf{R}^2$ par $S(x, v, \mu) = (G_v(x, v, \mu), G_\mu(x, v, \mu))$. Cette fonction est continue et dérivable si $b < \mu < a$ et $0 < x < 1$ et en vertu de la proposition 5.3 on a $S(x^*, v^*, \mu^*) = (0, 0)$. Si on dénote par $J_{v,\mu}$ le jacobien par rapport à v, μ on a

$$J_{v,\mu}(S) = \mathrm{Hess}_{v,\mu}(G)$$

et par la proposition 5.2 cette matrice est inversible en (x^*, v^*, μ^*). Par le théorème des fonctions implicites il existe donc un voisinage W de x^* et des fonctions dérivables $v(x)$ et $\mu(x)$ telle que $v(x) = v^*$, $\mu(x) = \mu^*$ et $G(x, v(x), \mu(x)) = 0$ pour tout $x \in W$. Puisque x^* est un minimum pour cette fonction

$$\frac{d}{dx} G(x^*, v(x^*), \mu(x^*)) = 0,$$

ce qui implique

$$G_x(x^*, v(x^*), \mu(x^*)) + G_v(x^*, v(x^*), \mu(x^*)) \frac{d}{dx} v(x) \Big|_{x=x^*}$$

$$+ G_\mu(x^*, v(x^*), \mu(x^*)) \frac{d}{dx} \mu(x) \Big|_{x=x^*} = 0$$

et donc $G_x(x^*, v(x^*), \mu(x^*)) = 0$ car les dérivées de G par rapport à v et μ sont nulles en $v(x^*)$ et $\mu(x^*)$. On conclut que si (x^*, v^*, μ^*) est une solution de (5.3) alors $G_x(x^*, v^*, \mu^*) = G_v(x^*, v^*, \mu^*) = G_\mu(x^*, v^*, \mu^*) = 0$. ∎

Remarque : puisque les contraintes $C(x)$ n'interviennent pas pour le maximum sur v, μ on aurait aussi pu démontrer cette proposition à l'aide de la condition nécessaire d'optimalité habituelle pour un problème min-max (voir Demyanov [9]).

La proposition 5.4 donne une condition nécessaire d'optimalité pour le cas $n = 1$. Pour généraliser cette condition à $n > 1$, considérons l'erreur sur un sous-intervalle $[x_{i-1}, x_{i+1}]$ comme au début de la section. Soit G_i l'analogue sur le sous-intervalle de la fonction G. Les formules (5.5) deviennent pour G_i

$$\begin{aligned}
\frac{\partial}{\partial x_i} G_i(x_i, v_i, \mu_i) &= \frac{m_{i-1}}{x_i - x_{i-1}} \int_{x_{i-1}}^{x_i} (t - x_{i-1}) h(t)\, dt \\
&\quad - \frac{m_i}{x_{i+1} - x_i} \int_{x_i}^{x_{i+1}} (t - x_{i+1}) h(t)\, dt - \mu_i \int_{P_{i-1}}^{P_i} h(t)\, dt, \\
\frac{\partial}{\partial v_i} G_i(x_i, v_i, \mu_i) &= \int_{P_{i-1}}^{P_i} h(t)\, dt - \frac{1}{x_i - x_{i-1}} \int_{x_{i-1}}^{x_i} (t - x_{i-1}) h(t)\, dt \\
&\quad - \frac{1}{x_{i+1} - x_i} \int_{x_i}^{x_{i+1}} (x_{i+1} - t) h(t)\, dt, \\
\frac{\partial}{\partial \mu_i} G_i(x_i, v_i, \mu_i) &= \int_{P_{i-1}}^{P_i} (t - x_i) h(t)\, dt,
\end{aligned} \tag{5.21}$$

où

$$m_i = \frac{v_{i+1} - v_i}{x_{i+1} - x_i}, \quad P_i = \frac{\mu_i x_i - \mu_{i+1} x_{i+1} + v_{i+1} - v_i}{\mu_i - \mu_{i+1}}. \tag{5.22}$$

Par les propositions 5.1 et 5.4, une condition nécessaire pour que (x, v, μ) soit une solution au problème (5.3) pour $n \geq 1$ est que les dérivées (5.21) soient nulles pour tout $i = 1, \ldots, n$. En écrivant pour le cas général $n \geq 1$ (où x, v, μ sont des vecteurs)

$$G(x, v, \mu) = \sum_{i=1}^{n+1} \int_{x_{i-1}}^{x_i} UL(x, v, \mu) h(t)\, dt$$

et en remarquant que seuls les termes i et $i+1$ de cette somme dépendent de x_i, v_i, μ_i, on peut calculer les dérivées de G par rapport à ces variables. Après simplifications on trouve que ces dérivées sont en fait données par les expressions (5.21), c'est-à-dire

que

$$G_{x_i} = \frac{\partial}{\partial x_i} G_i, \quad G_{v_i} = \frac{\partial}{\partial v_i} G_i, \quad G_{\mu_i} = \frac{\partial}{\partial \mu_i} G_i.$$

Par conséquent la condition nécessaire sur chaque sous-intervalle $[x_{i-1}, x_{i+1}]$ peut s'écrire simplement $G_{x_i} = G_{v_i} = G_{\mu_i} = 0$ pour tout i. On a donc :

Proposition 5.5 *Si (x^*, v^*, μ^*) est une solution au problème (5.3) alors*

$$\nabla G(x^*, v^*, \mu^*) = 0.$$

Preuve: Ceci découle de la discussion précédent la proposition. ∎

Nous avons ainsi déterminé une condition nécessaire d'optimalité pour le problème d'approximation L_h^1. Cette condition est simple au sens où il s'agit de trouver la solution à un système de $3n$ équations à $3n$ inconnues. Cependant ces équations sont non linéaires et lorsque n est grand le système peut être difficile à résoudre, surtout si la fonction de poids h est compliquée. Au chapitre suivant nous décrivons une méthode de solution reposant sur la décomposition du problème sur les sous-intervalles $[x_{i-1}, x_{i+1}]$.

5.3 Remarques sur la condition nécessaire

Le système $\nabla G(x, v, \mu) = 0$ peut être résolu numériquement mais cela exige des efforts qui peuvent être considérables. Il serait utile du point de vue pratique de simplifier ces équations pour accélérer la résolution. Nous présentons brièvement quelques idées qui pourraient donner des simplifications. Du point de vue théorique, les considérations ci-dessous permettent de mieux comprendre la nature du problème.

D'abord, posons

$$\alpha_i = \frac{1}{x_i - x_{i-1}} \int_{x_{i-1}}^{x_i} (t - x_{i-1}) h(t)\, dt,$$

$$\beta_i = \frac{1}{x_{i+1} - x_i} \int_{x_i}^{x_{i+1}} (x_{i+1} - t) h(t)\, dt.$$

Les quantités α_i et β_i ne dépendent que des x_i et sont strictement positives (car on suppose $x_i < x_{i+1}$ pour tout i). Le système $\nabla G(x, v, \mu) = 0$ s'écrit maintenant

$$\begin{cases} m_{i-1}\alpha_i + m_i\beta_i - \mu_i \displaystyle\int_{P_{i-1}}^{P_i} h(t)\, dt = 0 \\ \displaystyle\int_{P_{i-1}}^{P_i} h(t)\, dt - \alpha_i - \beta_i = 0 \\ \displaystyle\int_{P_{i-1}}^{P_i} (t - x_i) h(t)\, dt = 0 \end{cases} \tag{5.23}$$

pour $i = 1, \ldots, n$. Les deux premières équations de (5.23) impliquent que

$$\mu_i = \frac{m_{i-1}\alpha_i + m_i\beta_i}{\alpha_i + \beta_i}$$

donc les pentes μ_i sont uniquement déterminées par les x_i et les v_i. Ceci permet de réduire de n le nombre de variables.

Nous montrons maintenant qu'à l'optimalité les points P_i sont complétement déterminés par les x_i. La deuxième équation de (5.23) implique

$$\int_0^{P_i} h(t)\, dt = \alpha_i + \beta_i + \int_0^{P_{i-1}} h(t)\, dt \tag{5.24}$$

donc

$$\int_0^{P_i} h(t)\,dt = \sum_{k=1}^{i}(\alpha_k + \beta_k) + \int_0^{P_0} h(t)\,dt. \tag{5.25}$$

Par conséquent, pour x fixé, chaque P_i avec $i \geq 1$ est uniquement déterminé par P_0 car le membre de gauche de (5.25) est strictement croissant en P_i. Considérons maintenant les deux dernières équations de (5.23) pour $i = 1$:

$$\begin{cases} \displaystyle\int_{P_0}^{P_1} h(t)\,dt & = & \alpha_1 + \beta_1 \\ \\ \displaystyle\int_{P_0}^{P_1} (t - x_1)h(t)\,dt & = & 0. \end{cases} \tag{5.26}$$

Comme avant, la première des équations (5.26) implique que P_1 est uniquement déterminé par P_0 et donc on peut écrire $P_1 = g(P_0)$ pour une fonction continue g. De plus, si P_0 augmente alors puisque $h(t) > 0$, P_1 doit aussi augmenter dans le membre de gauche de cette équation, donc g est une fonction croissante de P_0. La deuxième équation de (5.26) s'écrit

$$\int_{x_1}^{g(P_0)} (t - x_1)h(t),\,dt = \int_{P_0}^{x_1} (x_1 - t)h(t)\,dt$$

où le membre de droite est décroissant en P_0 tandis que le membre de gauche est croissant. S'il y a une solution à cette équation, elle est donc unique. On a donc montré que la solution aux équations (5.26) est unique et ceci combiné avec (5.24) montre que les P_i sont uniquement déterminés en fonction des x_i.

En utilisant la définition (5.22) des P_i on obtient pour chaque $i = 0, \ldots, n$ une équation linéaire en v, μ

$$v_i - v_{i+1} + \mu_i(P_i - x_i) + \mu_{i+1}(x_{i+1} - P_i) = 0.$$

De plus, les première et deuxième équations de (5.23) donnent n équations supplémentaires

$$(\alpha_i + \beta_i)\mu_i = m_{i-1} + m_i\beta_i.$$

Au total, une solution à $\nabla G(x, v, \mu) = 0$ doit satisfaire $2n + 1$ équations linéaires à $2n$ inconnues v_i, μ_i. Ceci implique que les x_i ne sont pas indépendants et doivent être solution d'une équation $S(x_1, \ldots, x_n) = 0$, qui définit une surface dans $\mathbf{R}^n$. On voit donc que les x qui sont candidats pour la solution forment un sous-ensemble de $[0, 1]^n$ qui est de mesure nulle.

Nous avons présenté ces quelques idées pour amorcer une étude plus poussée de la condition nécessaire d'optimalité qui, nous espérons, débouchera sur une meilleure méthode de résolution.

CHAPITRE 6

ALGORITHMES OPTIMAUX

Nous présentons dans ce chapitre des algorithmes optimaux, séquentiellement optimaux et optimaux à chaque étape pour les problème d'approximation L^1, L^∞ et L_h^1, qui sont à notre avis les plus importants. Ces algorithmes seront ensuite comparés au chapitre 7.

6.1 Schémas généraux

Les problèmes L^1, L^∞ et L_h^1 ont la particularité que l'erreur totale sur [0,1] se décompose en erreurs partielles sur les sous-intervalles $[x_i, x_{i+1}]$ déterminés par les points d'évaluation. L'erreur totale est la somme des erreurs partielles pour L^1 et L_h^1 et le maximum des erreurs partielles pour L^∞. Ceci nous permet d'établir un schéma général pour un algorithme séquentiellement optimal et optimal à chaque étape pour ces problèmes.

6.1.1 Algorithme séquentiellement optimal

Un algorithme séquentiellement optimal se divise en deux étapes à chaque itération : d'abord le calcul de la meilleure répartition des points et ensuite le choix d'un point d'évaluation selon cette répartition. Généralisant Sukharev [38] un algorithme SO pour problèmes L^1, L^∞ et L_h^1 se définit comme suit.

Si i points d'évaluation $x_1^*, \ldots, x_i^*$ ont été choisis alors l'intervalle [0,1] est subdivisé en $i+1$ sous intervalles $[x_j^*, x_{j+1}^*]$, $j = 0, \ldots, i$. Comme à la section 1.8.1 dénotons

par z^i la situation réalisable définie par la valeur de la fonction à approximer et de son surgradient aux points x_j^*. Sur chaque sous-intervalle ces valeurs fixées aux extrémités définissent un problème d'approximation sur $[x_j^*, x_{j+1}^*]$. Soit $A_j(z^i, n_j)$ l'erreur optimale pour le problème sur le j^e intervalle si $m_j \leq n - i$ évaluations sont permises sur cet intervalle. La forme de A_j dépend de la norme employée pour mesurer l'erreur. La répartition optimale des $n-i$ points restants est celle qui réduit au maximum l'erreur $E(z^i, m_1, \ldots, m_{i+1})$ où

$$E(z^i, m_1, \ldots, m_{i+1}) = \sum_{j=1}^{i+1} A_j(z^i, m_j)$$

avec $\sum_{j=1}^{i+1} m_j = n - i$ pour l'approximation L^1 et L_h^1 et

$$E(z^i, m_1, \ldots, m_{i+1}) = \max_{j=1,\ldots,i+1} \{A_j(z^i, m_j)\}$$

avec $\sum_{j=1}^{i+1} m_j = n - i$ pour L^∞. L'algorithme SO procède comme suit.

Algorithme SO

<u>Itération 0</u>

1. (1) Poser $z^0 = (0, 1, f(0), f(1), f'(0), f'(1))$
   (2) Calculer les points optimaux $x_{11}^*, \ldots, x_{1n}^*$ sur [0,1] pour le problème déterminé par z^0
2. (1) Choisir un point x_{1j}^* et le dénoter par x_1^*
   (2) Calculer $f(x_1^*), f'(x_1^*)$
   (3) Poser $z^1 = (0, 1, f(0), f(x_1^*), f(1), f'(0), f'(x_1^*), f'(1))$

<u>Itération i</u>

L'intervalle [0,1] est subdivisé en $i + 1$ sous-intervalles par les i points déjà choi-

sis $x_1^*, \ldots, x_i^*$.

1. (1) Calculer la solution $(m_1^*, \ldots, m_i^*)$ au problème

$$\min_{m_j} E(z^i, m_1, \ldots, m_{i+1})$$

$$\text{s.c.} \quad \sum_{j=1}^{i+1} m_j = n - i$$
$$m_j \in \mathbf{N} \quad \forall j$$

   (2) Choisir un sous-intervalle j_0 pour lequel $m_{j_0}^* \neq 0$

2. (1) Calculer les $n_{j_0}^*$ points optimaux sur le sous-intervalle j_0 pour le problème déterminé par z^i

   (2) Choisir l'un des points trouvés en (1), dénoté x_{i+1}^*

   (3) Calculer $f(x_{i+1}^*)$ et $f'(x_{i+1}^*)$

   (4) Si nécessaire réordonner les indices des x_j^* de façon à ce que les points soient en ordre croissant et poser

$$x^{i+1} = (0, x_1^*, \ldots, x_{i+1}^*, 1)$$

$$v^{i+1} = (f(0), f(x_1^*), \ldots, f(x_{i+1}^*), f(1))$$

$$\mu^{i+1} = (f'(0), f'(x_1^*), \ldots, f'(x_{i+1}^*), f'(1))$$

$$z^{i+1} = (x^{i+1}, v^{i+1}, \mu^{i+1}).$$

   L'algorithme s'arrête lorsque $i = n - 1$.

Pour l'étape 1 il faut avoir une procédure pour calculer $E(z^i, m_1, \ldots, m_{i+1})$ pour toutes les valeurs de z^i et m_j, et c'est là la diffculté de la mise en oeuvre de cet algorithme. Nous avons une telle méthode pour l'approximation L^1 et L_h^1 mais pas

pour L^p, $p > 1$. En fait nous avons une formule explicite seulement dans le cas L^1 : on a vu au chapitre 3 (formule (3.10)) que

$$A_j(z^i, m_j) = \frac{a_j}{(m_j + 1)^2} \tag{6.1}$$

où a_j est l'erreur sur l'intervalle j si aucune évaluation n'y est faite, qui ne dépend que de z^i.

Pour L^1_h il faut utiliser une procédure numérique pour résoudre le système

$$\nabla G(x, v, \mu) = 0,$$

ce qui peut être coûteux si n est grand. C'est pourquoi nous avons choisi dans ce cas de remplacer le calcul exact de A_j par une approximation en utilisant la même formule (6.1) que pour le cas L^1, avec cette fois a_j mesurée dans la norme pondérée.

Lorsque A_j est de la forme (6.1) le programme en nombres entiers de l'étape 1 peut être résolu exactement avec la méthode de Gross [13] décrite ci-dessous, qui est basée sur le théorème suivant :

Théorème 6.1 *[13] Soit ϕ_j, $j = 1, \ldots, K$, des fonctions convexes et m un entier positif. Alors $(m_1, \ldots, m_K)$ est solution du problème en nombres entiers*

$$\min_{m_j} \quad \sum_{j=1}^{K} \phi_j(m_j)$$

$$\text{s.c.} \quad \sum_{j=1}^{K} m_j = M$$

$$m_j \in \mathbf{N} \ \ \forall j$$

si et seulement si

$$\min_{j=1,\ldots,K} \phi_j(m_j+1) - \phi_j(m_j) \geq \max_{\substack{j=1,\ldots,K \\ m_j \neq 0}} \phi_j(m_j) - \phi_j(m_j-1).$$

Le problème peut être résolu itérativement de la façon suivante :

Algorithme de résolution du PNE [13]

Itération 0

Poser $m_j = 0$, $j = 1, \ldots, K$.

Pour $1 \leq k \leq M$:

Itération k

(1) $\nu \in \underset{j=1,\ldots,K}{\operatorname{argmin}} \phi_j(m_j+1) - \phi_j(m_j)$

(2) $m_\nu \leftarrow m_\nu + 1$.

Plus de détails ainsi qu'une preuve d'optimalité sont donnés dans Saaty [31]. La complexité combinatoire de la procédure ci-dessus est d'ordre $\mathcal{O}(MK)$. On peut aussi pour réduire la complexité utiliser l'heuristique consistant à arrondir à la solution réalisable entière la plus proche la solution à la relaxation continue du problème, que l'on peut trouver facilement lorsque A_j est de la forme (6.1) : les x_j optimaux sont

$$x_j = \frac{(2a_j)^{\frac{1}{3}}}{\sum_{j=1}^{N}(2a_j)^{\frac{1}{3}}}.$$

Dans le schéma de l'algorithme SO l'intervalle à subdiviser et le point d'évaluation sur cet intervalle ne sont pas précisés puisqu'on ne peut définir de choix optimal. Selon Sukharev [40] on peut les déterminer aléatoirement puisqu'en théorie aucun choix n'est meilleur qu'un autre dans le pire cas. En pratique cependant nous avons constaté que les choix suivants donnent de meilleurs résultats et seront utilisés pour les tests du chapitre 7 :

- l'intervalle j_0 sur lequel l'erreur a priori est la plus grande, c'est-à-dire tel que

$$A_{j_0}(z^i, 0) = \max_{j=1,\ldots,i+1} \left\{ A_j(z^i, 0) \right\};$$

- sur l'intervalle j_0 le point d'évaluation du «milieu», c'est-à-dire celui d'indice $\lfloor (m^*_{j_0} + 1)/2 \rfloor$.

6.1.2 Algorithme optimal à chaque étape

Pour nos problèmes d'approximation un algorithme optimal à chaque étape est semblable à l'algorithme du sandwich de la section 4.3. La seule différence réside dans le choix de l'intervalle à subdiviser et du point d'évaluation. Plutôt que l'intervalle où l'erreur a priori est la plus importante, on subdivise celui où la diminution de l'erreur est la plus grande si un point y est placé. Le point choisi sur cet intervalle est le point optimal, c'est-à-dire celui qui réalise $A_j(z^i, 1)$. Cet algorithme procède donc comme suit :

Algorithme OSO

<u>Itération 0</u>

Poser $z^1 = (0, 1, f(0), f(1), f'(0), f'(1))$

Itération i

L'intervalle [0,1] est subdivisé en $i+1$ sous-intervalles par les i points déjà choisis $x_1^*, \ldots, x_i^*$.

1. Trouver un sous-intervalle j_0 tel que

$$A_{j_0}(z^i, 1) - A_{j_0}(z^i, 0) = \max_{j=1,\ldots,i+1} \left\{ A_j(z^i, 1) - A_j(z^i, 0) \right\}$$

2. Déterminer $x_i^* \in [x_{j_0}^*, x_{j_0+1}^*]$ qui réalise le minimum $A_{j_0}(z^i, 1)$
3. Calculer $f(x_i^*)$, $f'(x_i^*)$
4. Poser

$$\begin{aligned} x^{i+1} &= (0, x_1^*, \ldots, x_{i+1}^*, 1) \\ v^{i+1} &= (f(0), f(x_1^*), \ldots, f(x_{i+1}^*), f(1)) \\ \mu^{i+1} &= (f'(0), f'(x_1^*), \ldots, f'(x_{i+1}^*), f'(1)) \\ z^{i+1} &= (x^{i+1}, v^{i+1}, \mu^{i+1}). \end{aligned}$$

L'algorithme s'arrête lorsque $i = n - 1$.

Cet algorithme est beaucoup plus simple à mettre en oeuvre que l'algorithme SO puisque $A_j(z^i, 1)$ est souvent facile à calculer et qu'il n'y a pas de programme en nombre entiers à résoudre à chaque itération. De plus l'algorithme OSO pour l'approximation L^p est d'ordre optimal puisque l'algorithme du sandwich l'est (voir section 4.3). Sukharev [40] remarque qu'en général un algorithme OSO peut donner de très mauvais résultats mais les expériences montrent que pour nos problèmes ce n'est pas le cas.

Nous décrivons maintenant en détails ces algorithmes pour les problèmes L^1, L^∞ et L_h^1.

6.2 Approximation L^1

Les algorithmes pour l'approximation L^1 sont basés sur les formules pour les points d'évaluation et l'erreur optimaux proposés par Sonnevend [33] et Guérin et al. [14].

6.2.1 Algorithme optimal

Soit n le nombre de points d'évaluation et a, b les dérivées en 0 et 1. L'algorithme optimal passif est donné par le choix suivant des points d'évaluation :

$$x_i = \frac{i}{(n+1)^2}\left[i + 2(n+1-i)\left(\frac{1-b}{a-b}\right)\right]$$

pour $i = 1, \ldots, n$.

La complexité (combinatoire) de la procédure qui résulte de cette formule est de $\mathcal{O}(n)$.

L'adaptation n'est pas utile ici mais la récurrence de la section 3.3 et sa solution donnent une autre méthode, adaptative, pour calculer les points d'approximation. Cette méthode est intéressante et elle est considérée ici parce qu'elle est très simple et est en fait du même ordre de complexité que l'algorithme passif. Elle est décrite en détails dans Guérin [15]. Ces deux algorithmes seront comparés au chapitre 7.

6.2.2 Algorithme séquentiellement optimal

Dans le schéma général de la section 6.1.1 si $x^i = (x_1, \ldots, x_i)$, $v^i = (v_1, \ldots, v_i)$, $\mu^i = (\mu_1, \ldots, \mu_i)$ et $z^i = (x^i, v^i, \mu^i)$ alors :

- Par la formule (3.9) de la section 3.3

$$A_j(z^i, m_j) = \frac{[\mu_j(x_{j+1} - x_j) - (v_{j+1} - v_j)][v_{j+1} - v_j - \mu_{j+1}(x_{j+1} - x_j)]}{2(m_j + 1)^2(\mu_j - \mu_{j+1})}.$$

- La méthode exacte de Gross est employée pour la solution du programme en nombres entiers de l'étape 1. L'algorithme SO qui en résulte possède une complexité combinatoire d'ordre $\mathcal{O}(n^2)$. Pour un algorithme SO du même ordre de complexité que l'algorithme passif optimal, soit $\mathcal{O}(n)$, on peut plutôt employer l'heuristique d'arrondi discutée ci-dessus en 6.1.1.

Nous n'avons pas considéré d'algorithme optimal à chaque étape pour ce problème car l'algorithme SO est supérieur et est déjà très simple.

6.3 Approximation L^∞

Le problème d'approximation L^∞ est difficile et on ne connaît pas d'algorithme de solution atteignant la borne inférieure $\inf_{\alpha \in \hat{A}} e(\alpha)$. Pour cette raison, il n'est pas possible de calculer exactement les $A_j(z^i, m_j)$ lorsque $m_j > 1$ et donc de définir un algorithme séquentiellement optimal. Cependant nous proposons ci-dessous un algorithme optimal à chaque étape pour ce problème.

6.3.1 Algorithme optimal à chaque étape

Le cas $n = 1$ a été examiné à la section 4.4 et on a vu qu'on peut trouver une solution analytique à l'aide de la proposition 4.4. L'algorithme OSO pour ce problème est ensuite défini comme dans le schéma général de la section 6.1.2.

$A_j(z^i, 1)$ est calculé comme suit :

1. Par un changement de variables, ramener la situation sur $[x_j^*, x_{j+1}^*]$ à la situation standard sur [0,1].
2. Calculer le point x optimal en minimisant la fonction $\theta(x)$ de la proposition 4.4.
3. En utilisant le changement de variables inverse, calculer l'erreur et le point optimal sur $[x_j^*, x_{j+1}^*]$.

6.4 Approximation L_h^1

Les algorithmes de cette section sont basés sur la condition nécessaire (5.5) de la section 5.2.

6.4.1 Algorithme optimal

L'algorithme optimal passif est déterminé par les points d'évaluation x_i qui sont solution de $\nabla G(x, v, \mu) = 0$. Sur chaque sous-intervalle $[x_{i-1}, x_{i+1}]$ ceci donne le système de trois équations et trois inconnues x_i, v_i, μ_i, $\nabla G_i(x_i, v_i, \mu_i) = 0$. La procédure suivante cherche à résoudre $\nabla G(x, v, \mu) = 0$ en procédant une variable à la fois (UVALF) sur chaque sous-intervalle. Nous dénotons par f_s la fonction pire cas pour le problème d'approximation L^1, définie par Sonnevend [33].

Algorithme UVALF

Itération 0

1. Choisir $\epsilon_r > 0$, $\epsilon_d > 0$.
2. Soit $x_{1j} = \dfrac{j}{n+1}$, $j = 0, \ldots, n+1$.
3. Calculer $v_{1j} = f_s(x_{1j})$, $\mu_{1j} = f_s'(x_{1j})$ pour $j = 0, \ldots, n+1$.

4. Poser $x^1 = (x_{10}, \ldots, x_{1(n+1)})$, $v^1 = (v_{10}, \ldots, v_{1(n+1)})$, $\mu^1 = (\mu_{10}, \ldots, \mu_{1(n+1)})$.

Itération k

1. Soit $z^k = (x^k, v^k, \mu^k)$ les valeurs courantes pour les variables.
2. Pour chaque sous-intervalle $[x_{k(i-1)}, x_{k(i+1)}]$, $i = 1, \ldots, n$.
   (i) Fixer tous les x_j, v_j, μ_j à leur valeur courante sauf x_i, v_i, μ_i.
   (ii) Trouver la solution x_i^*, v_i^*, μ_i^* aux équations $\nabla G_i(x_i, v_i, \mu_i) = 0$.
   (iii) Poser $x_{(k+1)i} \leftarrow x_i^*$, $v_{(k+1)i} \leftarrow v_i^*$, $\mu_{(k+1)i} \leftarrow \mu_i^*$.
3. Soit $z^{k+1} = (x^{k+1}, v^{k+1}, \mu^{k+1})$. Calculer

$$\text{dist}_k = ||x^{k+1} - x^k||$$

$$\text{res}_k = \max_{i=1,\ldots,n} || \nabla G_i(x_{(k+1)i}, v_{(k+1)i}, \mu_{(k+1)i}) ||_\infty.$$

Si

$$\text{res}_k < \epsilon_r \text{ ou } \text{dist}_k < \epsilon_d$$

alors la solution optimale cherchée est x^{k+1}. Sinon, poser $k \leftarrow k+1$ et retourner à 1.

L'algorithme s'arrête lorsque les équations sur chaque intervalle sont résolues à ϵ_r près ou que d'une itération à la suivante les points x^{k+1} et x^k sont à moins de ϵ_d l'un de l'autre. Ce qui nous intéresse à la conclusion de l'algorithme ce sont les points x_i et c'est pourquoi l'algorithme s'arrête lorsque la partie en x de la solution aux équations semble avoir convergé, même si celles-ci n'ont pas encore été résolues avec la marge d'erreur prescrite.

L'algorithme UVALF est semblable à la méthode de Gauss-Seidel décrite dans Burden [4] pour la solution d'un système d'équations linéaires. En pratique nous avons

constaté que UVALF convergeait vers une solution aux équations dans tous les cas mais nous n'avons pas fait ici d'analyse de convergence détaillée.

6.4.2 Algorithme séquentiellement optimal

Dans le schéma général les $A_j(z^i, m_j)$ devraient être calculés à l'aide de la procédure UVALF. Cependant l'algorithme serait alors complexe et exigerait beaucoup de calculs. En particulier, pour évaluer $A_j(z^i, m_j)$ à la i^e itération il ferait appel à UVALF

$$\begin{pmatrix} n-i \\ m_1, m_2, \ldots, m_i \end{pmatrix}$$

fois. Nous avons choisi, pour les tests du chapitre 7, d'apporter la simplification suivante à la procédure SO : lors de l'étape 1, remplacer le calcul exact des $A_j(z^i, m_j)$ par l'approximation $\frac{a_j}{(n+1)^2}$, où

$$a_j = \int_{x_j}^{x_{j+1}} [U(z^i, t) - L(z^i, t)]h(t)\, dt. \tag{6.2}$$

Dans le schéma général on a donc :

- À l'étape 1, $A_j(z^i, m_j)$ est approximé par $\frac{a_j}{(n+1)^2}$ tel que défini par (6.2) et le PNE résolu avec l'algorithme de Gross.
- À l'étape 2, les points optimaux sur l'intervalle à subdiviser sont calculés à l'aide de la procédure UVALF.

6.4.3 Algorithme optimal à chaque étape

Dans le schéma général, $A_j(z^i, 1)$ est calculé en résolvant le système de trois équations et trois inconnues $\nabla G_j(x, v, \mu) = 0$ sur $[x_j^*, x_{j+1}^*]$.

CHAPITRE 7

RÉSULTATS NUMÉRIQUES

Nous présentons dans ce chapitre les résultats d'expériences numériques faites sur les différents algorithmes présentés au chapitre précédent. Le but de ces expériences est, d'abord, de montrer qu'il est possible d'implanter concrètement ces algorithmes et ce de manière assez simple. Ensuite, de comparer les différentes méthodes d'approximation et d'évaluer leurs mérites relatifs.

7.1 Méthodologie des tests

Chaque algorithme a été testé sur des échantillons de fonctions concaves. Ces fonctions sont de trois types : lisses (C1), lisses par morceaux (C0), c'est-à-dire que la dérivée peut être discontinue, et linéaires par morceaux (LPM). Dans cette section, une fonction f est dite *normalisée* si $f(0) = 0$ et $f(1) = 1$.

Chaque échantillon contient 30 fonctions normalisées et a été produit à l'aide des procédures décrites à la section 7.2. Dans la suite, lorsqu'un paramètre est choisi aléatoirement sur un intervalle donné, le choix sera toujours fait selon une distribution uniforme sur cet intervalle. Les dérivées aux extrémités $a = f'(0)$ et $b = f'(1)$ ont été choisies aléatoirement sur les intervalles $1 \leq a \leq 100$ et $-100 \leq b \leq 1$. L'échantillon obtenu possède les caractéristiques données au tableau 7.1.

Pour chaque fonction test l'erreur d'approximation pour un algorithme donné a été calculée pour $n = 1, \ldots, 12$ points d'évaluation. L'erreur a été ensuite divisée par l'erreur a priori pour cette fonction, c'est-à-dire l'erreur maximum étant donnée

Tableau 7.1 Valeurs extrêmes des paramètres pour les fonctions tests

	a min	a max	b min	b max
C1	1.58	98.79	-96.76	0.15
C0	3.55	97.86	-99.91	-1.68
LPM	1.34	96.26	-97.51	0.83

seulement les dérivées aux extrémités, pour obtenir une erreur relative. Ceci permet de compenser les différences entre les normes des différentes fonctions. Enfin, la moyenne des erreurs relatives a été calculée. Pour mieux visualiser les comparaisons entre les algorithmes, nous avons également représenté sur un graphe le rapport des erreurs en fonction du nombre de points d'évaluation. Les rapports sont de la forme

$$\frac{\text{algorithme de base}}{\text{algorithme}}$$

où l'algorithme de base est celui qui est le plus simple pour un problème donné. Les détails de ces comparaisons se trouvent dans les sections 7.3 à 7.5.

7.2 Fonctions tests

Les fonctions utilisées pour les tests sont construites par morceaux à partir de polynômes de façon à ce que la fonction résultante soit lisse, continue ou linéaire par morceaux, selon le cas. Cette construction est semblable à la méthode d'interpolation par des splines cubiques décrite dans Fortin [10].

7.2.1 Fonctions lisses, première méthode.

Soit $p_k = c_{k3}x^3 + c_{k2}x^2 + c_{k1}x + c_{k0}$, $k = 1, 2$, deux polynômes de degré trois, $x_1 \in [0, 1]$ et $\gamma_0, \gamma_1, \gamma_2$ des nombres négatifs. On choisit les coefficients c_{kl} de façon à ce que

1. $p_1(0) = 0$, $p_2(1) = 1$ (normalisation)
2. $p_1(x_1) = p_2(x_1)$ (continuité)
3. $p_1'(x_1) = p_2'(x_1)$ (continuité de f')
4. $p_1''(0) = \gamma_0$, $p_1''(x_1) = \gamma_1$, $p_2''(x_1) = \gamma_1$, $p_2''(1) = \gamma_2$ (concavité).

Ces huit équations linéaires permettent de déterminer les huit coefficients c_{kl}. En choisissant aléatoirement $x_1 \in [0, 1]$ et $\gamma_i \in [-300, 0]$, on obtient une fonction concave normalisée en définissant

$$f(x) = \begin{cases} p_1(x) & \text{si } x \in [0, x_1] \\ \\ p_2(x) & \text{si } x \in [x_1, 1]. \end{cases}$$

Notons que cette méthode pourrait être généralisée en utilisant $k \geq 2$ polynômes.

7.2.2 Fonctions lisses, deuxième méthode.

Si de plus on exige que les dérivées de f soient fixées aux extrémités, disons $f'(0) = a, f'(1) = b$, on peut procéder comme suit.

Aux équations 1 à 4 ci-dessus on ajoute

$$p_1'(0) = a, \quad p_2'(1) = b.$$

Pour que ce système à dix équations ait une solution il faut imposer des contraintes à x_1 et aux γ_i. Explicitement, les deux équations supplémentaires sont

$$c_{11} = a, \quad \text{et} \quad 3c_{22} + 2c_{21} + c_{20} = b. \tag{7.1}$$

Lorsque les coefficients c_{kl} sont exprimés en termes de x_1 et γ_i à l'aide de la solution

aux équations 1-4, les équations (7.1) deviennent

$$\begin{aligned} 6 + x_1^2(\gamma_0 - \gamma_2) + x_1(\gamma_1 + 2\gamma_2 - 3\gamma_0) - (\gamma_1 + \gamma_2) &= 6a \\ 6 + x_1^2(\gamma_0 - \gamma_2) + x_1(\gamma_1 - \gamma_2) + 2\gamma_2 + \gamma_1 &= 6b. \end{aligned}$$

Résolvant pour γ_1, γ_2 on trouve

$$\begin{aligned} \gamma_1 &= -2x_1\gamma_0 - 2x_1(a-b) - 4(a-1) + 2(1-b) \\ \gamma_2 &= x_1\gamma_0 + 2x_1(a-b) + 2(a-1) - 4(1-b). \end{aligned} \tag{7.2}$$

Pour que f soit concave il faut que $\gamma_1, \gamma_2 \leq 0$, ce qui est possible seulement si

$$3\frac{1-b}{a-b} - 2 \leq x_1 \leq 3\frac{1-b}{a-b}. \tag{7.3}$$

Dans ce cas il faut aussi que $\gamma_0 \leq 0$ satisfasse

$$-(a-b) - \frac{2}{x_1}(a-1) + \frac{1}{x_1}(1-b) \leq \gamma_0 \leq -2(a-b) + \frac{4}{x_1}(1-b) - \frac{2}{x_1}(a-1). \tag{7.4}$$

Par conséquent pour générer f concave normalisée avec $f'(0) = a$ et $f'(1) = b$:

1. Choisir aléatoirement $x_1 \in [0, 1]$ satisfaisant (7.3)
2. Choisir ensuite aléatoirement $\gamma_0 \leq 0$ satisfaisant (7.4)
3. Calculer γ_1, γ_2 à l'aide de (7.2)
4. Calculer les coefficients c_{kl} en résolvant les équations 1-4.

7.2.3 Fonctions lisses par morceaux.

Pour générer une fonction f concave normalisée lisse par morceaux, c'est-à-dire dont la dérivée peut être discontinue en certains points :

1. Poser $x_0 = 0, x_{m+1} = 1$ et choisir aléatoirement m points $x_i \in]0,1[$ et v_i, μ_i correspondants de telle sorte que (x, v, μ) soit une situation réalisable, c'est-à-dire que $v, \mu \in C(x)$ (voir la définition (3.4)).
2. Sur chaque sous-intervalle $[x_i, x_{i+1}]$ générer une fonction concave lisse satisfaisant $f(x_i) = v_i, f(x_{i+1}) = v_{i+1}, f'(x_i) = \mu_i, f'(x_{i+1}) = \mu_{i+1}$ à l'aide de la deuxième méthode 7.2.2 et d'un changement de variable approprié.

7.2.4 Fonctions linéaires par morceaux.

Pour générer une fonction concave normalisée linéaire par morceaux :

1. Poser $x_0 = 0, x_{m+1} = 1$ et choisir aléatoirement m points $x_i \in]0,1[$ et v_i, μ_i correspondants de telle sorte que (x, v, μ) soit une situation réalisable, c'est-à-dire que $v, \mu \in C(x)$ (voir la définition (3.4)).
2. Sur $[x_i, x_{i+1}]$, définir

$$f(x) = \min\{\mu_i(x - x_i) + v_i, \mu_{i+1}(x - x_{i+1}) + v_{i+1}\}.$$

7.3 Approximation L^1 : OPT vs DYN vs SO

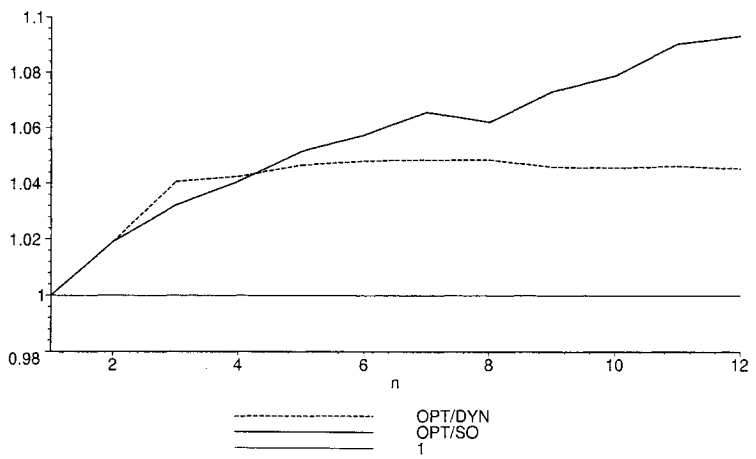
Nous comparons ici les algorithmes suivants pour l'approximation dans la norme L^1 :

1. Optimal (OPT), basé sur la méthode de Sonnevend [33].
2. Algorithme de programmation dynamique (DYN), tel que décrit à la section 3.3.
3. Séquentiellement optimal (SO), tel que décrit à la section 6.4.2.

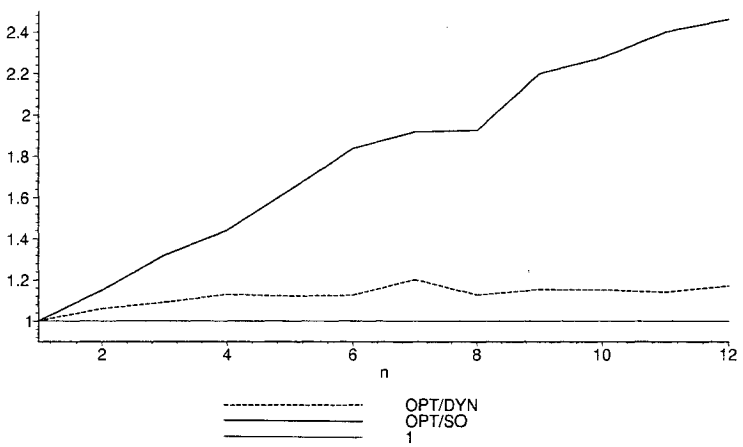
Le tableau 7.2 donne les erreurs moyennes pour les différents problèmes en fonction du nombre de points d'évaluation. L'algorithme de base est OPT et les graphes 7.3 ci-dessous représentent les rapports OPT/DYN et OPT/SO.

Tableau 7.2 Moyenne des erreurs d'approximation L^1.

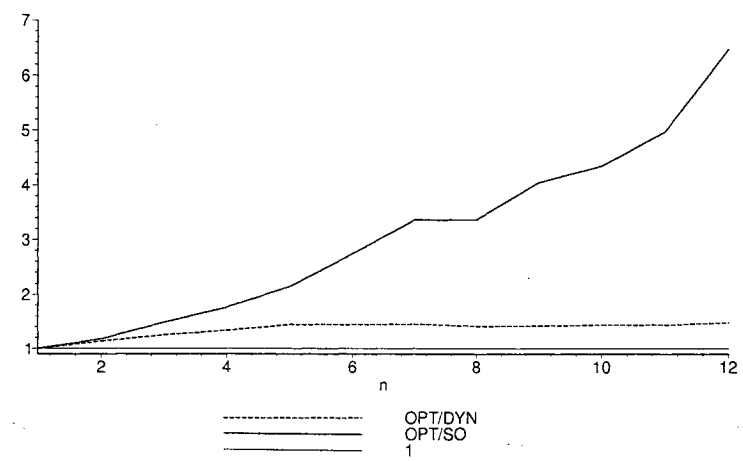
	Fonctions lissses			Fonctions C0			Fonctions LPM		
n	OPT	DYN	SO	OPT	DYN	SO	OPT	DYN	SO
1	.241711	.241711	.241711	.164692	.164692	.164692	.186173	.186173	.186173
2	.107096	.105086	.105086	.068794	.064904	.059800	.083571	.073845	.070999
3	.061082	.058705	.059174	.036713	.033672	.027793	.046555	.037454	.031489
4	.039074	.037484	.037547	.022916	.020300	.015900	.031246	.023545	.017775
5	.027233	.026024	.025899	.016659	.014872	.010169	.021949	.015311	.010257
6	.020018	.019102	.018931	.011969	.010636	.006514	.015742	.010985	.005751
7	.015346	.014638	.014401	.009678	.008057	.005046	.011698	.008091	.003485
8	.012121	.011562	.011413	.006973	.006191	.003622	.009213	.006557	.002739
9	.009809	.009380	.009141	.005977	.005191	.002721	.007193	.005098	.001781
10	.008113	.007760	.007521	.004918	.004276	.002162	.005517	.003870	.001271
11	.006829	.006528	.006263	.004161	.003652	.001736	.004511	.003164	.000909
12	.005828	.005576	.005331	.003509	.002997	.001427	.004026	.002729	.000623



(a) Fonctions lisses



(b) Fonctions lisses par morceaux



(c) Fonctions linéaires par morceaux

Figure 7.1 Comparaison de OPT, DYN et SO pour l'approximation L^1.

On voit qu'en pratique les méthodes adaptatives sont meilleures que la méthode optimale passive, en particulier pour les fonctions C0 et linéaires par morceaux. Pour les fonctions lisses DYN et SO sont comparables mais SO est nettement plus efficace pour les fonctions dont la dérivée est discontinue. Le coût des trois algorithmes OPT, DYN et SO pour le problème d'approximation L^1 est sembable et en fait ils ont la même complexité combinatoire, soit $\mathcal{O}(n)$, si l'heuristique d'arrondi est utilisée dans SO pour la résolution du problème en nombres entiers. Par conséquent, l'algorithme SO est à la fois efficace et peu coûteux pour le problème d'approximation L^1 et ce pour les trois types de fonctions testées ici.

7.4 Approximation L^∞ : SAND vs OSO

Dans cette section, nous comparons les algorithmes suivants pour l'approximation dans la norme L^∞ :

1. Algorithme du sandwich (SAND) de Burkard et al. [5].
2. Algorithme optimal à chaque étape (OSO) tel que décrit à la section 6.3.1.

Le tableau 7.3 donne les erreurs moyennes pour ces deux méthodes. L'algorithme de base ici est SAND et les graphes 7.4 ci-dessous représentent le rapport SAND/OSO.

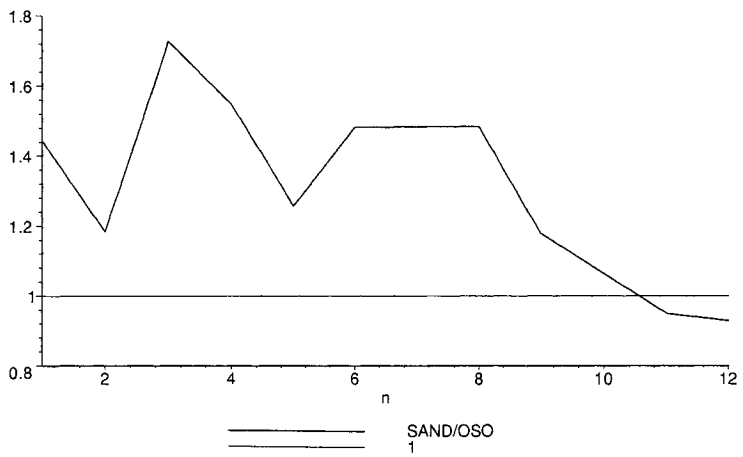
On constate que l'algorithme OSO est supérieur à l'algorithme du sandwich dans presque tous les cas. Pour les fonctions lisses SAND est meilleur pour $n = 11, 12$ mais il faudrait faire d'autres tests avec un échantillon plus grand et plus de points pour vérifier si SAND est effectivement supérieur que OSO pour de grandes valeurs de n.

De même, il serait intéressant de voir avec d'autres tests si le comportement en dents de scie du rapport DYN/OSO est caractéristique de ces algorithmes. Rote [30] a déjà constaté ce genre de variations pour l'algorithme du sandwich avec différentes règles de subdivision. Dans son cas, les expériences portaient sur des fonctions (convexes) lisses particulières : $\sqrt{x}$, $\sqrt{1-x^2}$ et $\sin x$ et sur un grand nombre de points ($n = 1024$). L'auteur a observé plusieurs types de comportements périodiques de l'erreur par rapport au nombre de points.

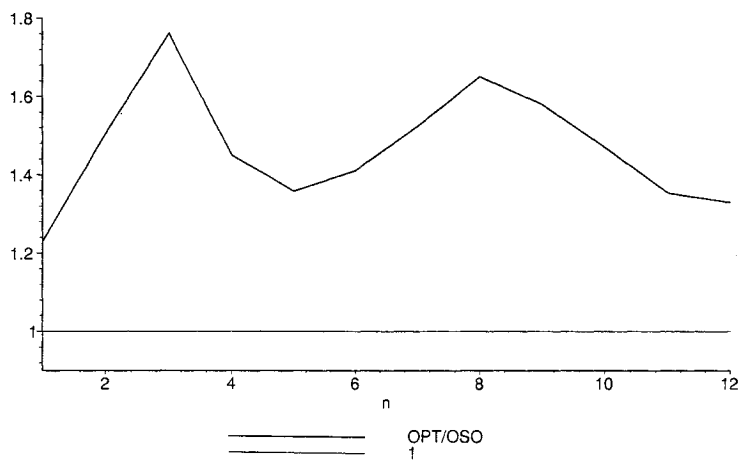
Malgré ces questions, et bien que l'algorithme OSO soit plus coûteux que SAND, il nous semble que le premier soit supérieur si les fonctions à approximer sont difficiles à évaluer et le nombre d'évaluations est petit.

Tableau 7.3 Moyennes des erreurs d'approximation L^∞.

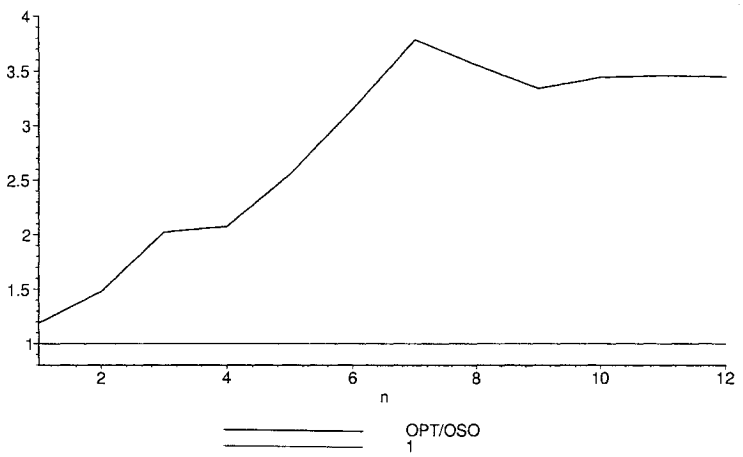
	Fonctions lissses		Fonctions C0		Fonctions LPM	
n	SAND	OSO	SAND	OSO	SAND	OSO
1	.424092	.293441	.321545	.261642	.319533	.268368
2	.268978	.227217	.170765	.113509	.199279	.134797
3	.152706	.088403	.098709	.056035	.126178	.062401
4	.108063	.069867	.051631	.035607	.082294	.039693
5	.079428	.063198	.032064	.023603	.063039	.024638
6	.062302	.042036	.023994	.017001	.048301	.015317
7	.041040	.027669	.018456	.012123	.038227	.010096
8	.027412	.018480	.014241	.008627	.025566	.007193
9	.021039	.017850	.011208	.007092	.015699	.004700
10	.018204	.017126	.008737	.005943	.012184	.003541
11	.015609	.016444	.006882	.005084	.009071	.002624
12	.013473	.014500	.005740	.004318	.006938	.002013



(a) Fonctions lisses



(b) Fonctions lisses par morceaux



(c) Fonctions linéaires par morceaux

Figure 7.2 Comparaison de SAND, et OSO pour l'approximation L^∞.

7.5 Approximation L_h^1 : OPT vs OSO vs SO

Dans cette section nous comparons les algorithmes suivants pour l'approximation dans la norme L_h^1 :

1. Algorithme optimal UVALF décrit à la section 6.4.1.
2. Algorithme optimal à chaque étape (OSO), où le calcul du point optimal sur chaque sous-intervalle est fait à l'aide de UVALF.
3. Algorithme séquentiellement optimal (SO), où les points optimaux sur chaque sous-intervalle sont calculés à l'aide de UVALF. À chaque étape, la répartition des points sur les sous-intervalles est obtenue avec l'heuristique discutée à la

section 6.4.2. Cet algorithme est donc en fait une approximation d'un algorithme SO.

En utilisant la notation de la section 6.4.1 le critère d'arrêt pour l'algorithme UVALF est le suivant : $\epsilon_r = 10^{-6}$ et $\epsilon_d = 10^{-6}$. L'algorithme s'arrête donc lorsque les équations définissant les points optimaux sont résolues à 10^{-6} près ou la variation des points optimaux est de moins de 10^{-6} d'une itération à l'autre.

Les algorithmes ont été testés avec deux fonctions de poids, soit $h_1(t) = t$ et $h_2(t) = t(1-t)$. En l'absence d'autres critères, ces fonctions ont été choisies parce que les équations correspondantes sont relativement simples. D'autres fonctions de différents types (exponentielles, linéaires par morceaux, discontinues, etc.) ont été employées avec certaines fonctions tests mais les calculs numériques avec Maple, bien que possibles, se sont avérés trop lourds pour des tests à plus grande échelle.

Les tableaux 7.5 et 7.5 donnent les erreurs moyennes pour les trois algorithmes. L'algorithme de base ici est OPT et les graphes 7.5 et 7.5 ci-dessous représentent les rapports OPT/OSO et OPT/SO.

La première conclusion à tirer de ces tests est que les méthodes adaptatives donnent en pratique de meilleurs résultats pour les fonction qui ne sont pas lisses. Pour les fonctions lisses, on constate ici aussi un comportement en dents de scie pour l'algorithme OSO, qui rappelle les variations périodiques de l'erreur observées par Rote [30]. L'algorithme SO, lui, présente un comportement plus régulier et est toujours supérieur à OPT et OSO.

Cependant pour les fonction C0 et linéaires par morceaux, OSO est beaucoup plus efficace que dans le cas lisse. Il semble donc que les méthodes OPT et SO puissent tirer un meilleur parti de l'information sur le nombre de points à choisir dans le cas de fonctions lisses mais que OSO réussit à très bien s'adapter aux fonctions qui ne

le sont pas.

Enfin on observe que OSO est presqu'aussi efficace et parfois même plus que SO pour ces dernières fonctions. Ceci est peut-être dû au fait que l'information sur le nombre de points est moins pertinente que la capacité à s'adapter lorsque les fonctions présentent de brusques variations de leur dérivée. La similarité des performances tient peut-être aussi au fait que l'algorithme SO testé n'est en fait qu'une approximation d'un algorithme séquentiellement optimal, dû à la simplification du sous-problème de la répartition des points discutée ci-dessus. Si c'est le cas, il est possible qu'un véritable algorithme SO donne de meilleurs résultats que OSO pour le problème d'approximation pondérée.

À ce propos, nous rappelons que pour l'approximation L_h^1 avec $h \equiv 1$ l'erreur e_n lorsque n évaluations sont faites est exactement $e_n = e_0/(n+1)^2$ et dans ce cas l'approximation de l'algorithme SO est en fait exacte. Pour les autres fonctions de poids, le rapport $r = e_n/(e_0/(n+1)^2)$ n'est pas constant mais il est le même quelque soit n. Le tableau 7.4 ci-dessous donne ce rapport pour les cinq premières fonctions tests lisses et $h(t) = t$.

Tableau 7.4 Rapport $r = e_n/(e_0/(n+1)^2)$ pour cinq fonctions lisses.

fonction	1	2	3	4	5
$f'(0)$	43.31	74.92	26.72	96.08	95.15
$f'(1)$	-67.56	-96.76	-68.68	-17.89	-85.20
r	1.8537	1.9115	1.7337	2.5734	2.0298

Nous n'avons pas su déterminer une relation entre r, $f'(0)$ et $f'(1)$. Une telle relation serait très utile car elle permettrait d'implanter facilement un algorithme SO pour le problème L_h^1 sans avoir recours à UVALF pour la résolution du PNE à l'étape 1. De

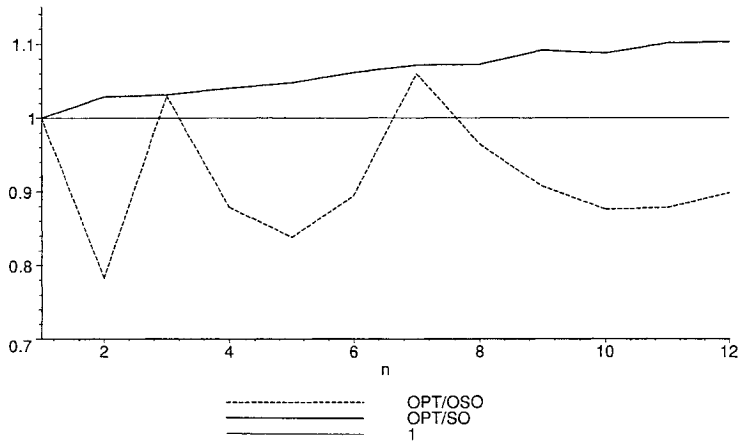
plus, connaissant la valeur optimale du problème de min-max il serait probablement plus simple de calculer les points optimaux.

Cependant si on s'en tient aux algorithmes testés il semble que OSO est préférable à SO pour les fonctions C0 et linéaires par morceaux puisque la légère différence d'efficacité est amplement compensée par le coût moindre de OSO.

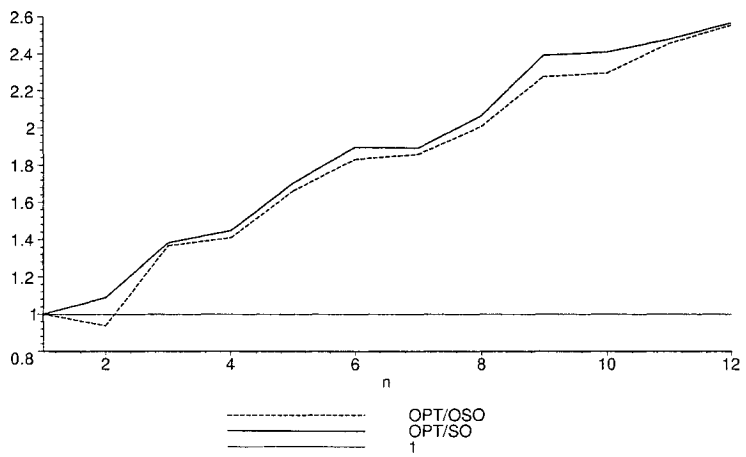
Fonction de densité $h_1(t) = t$.

Tableau 7.5 Moyennes des erreurs d'approximation L_h^1 avec $h(t) = t$.

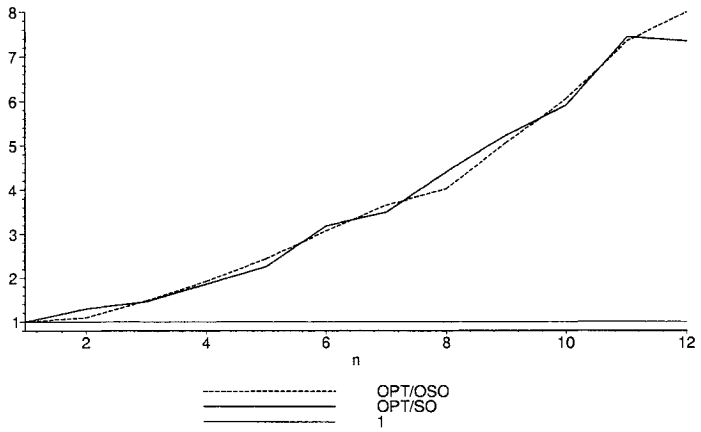
	Fonctions lissses			Fonctions C0			Fonctions LPM		
n	OPT	OSO	SO	OPT	OSO	SO	OPT	OSO	SO
1	.222889	.222889	.222889	.140108	.140108	.140108	.165509	.165509	.165509
2	.096367	.123169	.093678	.052229	.055851	.047925	.074055	.068213	.057087
3	.053217	.051677	.051595	.030859	.022584	.022300	.038957	.026332	.026800
4	.033763	.038448	.032452	.018057	.012822	.012461	.026672	.013918	.014348
5	.023343	.027873	.022276	.012780	.007700	.007500	.017902	.007382	.007957
6	.017089	.019112	.016096	.009617	.005259	.005071	.013042	.004257	.004100
7	.013036	.012308	.012165	.006883	.003710	.003640	.009898	.002718	.002837
8	.010281	.010675	.009585	.005504	.002742	.002663	.007344	.001829	.001665
9	.008327	.009181	.007629	.004753	.002090	.001987	.006195	.001227	.001186
10	.006878	.007857	.006324	.003767	.001642	.001563	.005016	.000831	.000850
11	.005776	.006578	.005244	.003159	.001287	.001274	.004330	.000590	.000582
12	.004916	.005477	.004457	.002667	.001045	.001039	.003482	.000436	.000474



(a) Fonctions lisses.



(b) Fonctions lisses par morceaux.



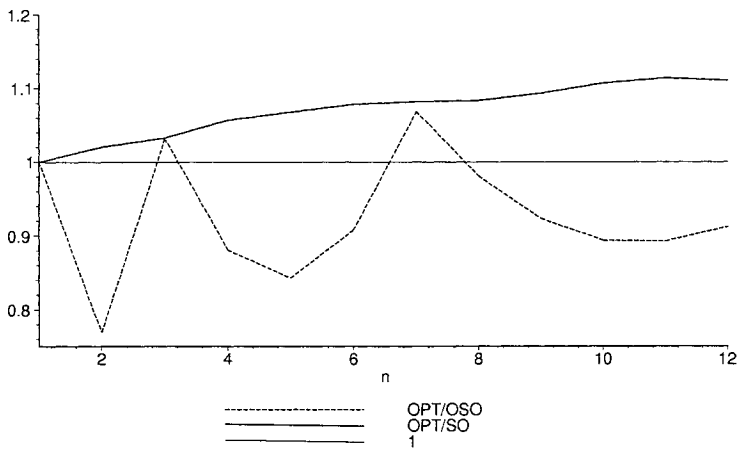
(c) Fonctions linéaires par morceaux.

Figure 7.3 Comparaison de OPT, OSO et SO pour l'approximation L_h^1 avec $h(t) = t$.

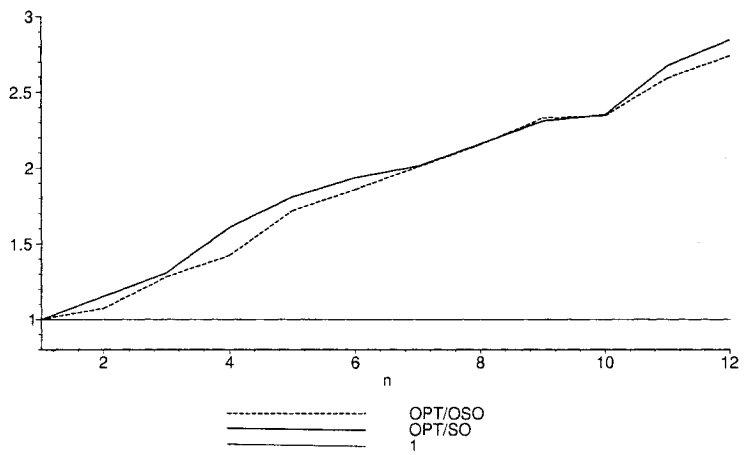
Fonction de densité $h_2(t) = t(1-t)$.

Tableau 7.6 Moyennes des erreurs d'approximation L_h^1 avec $h(t) = t(1-t)$.

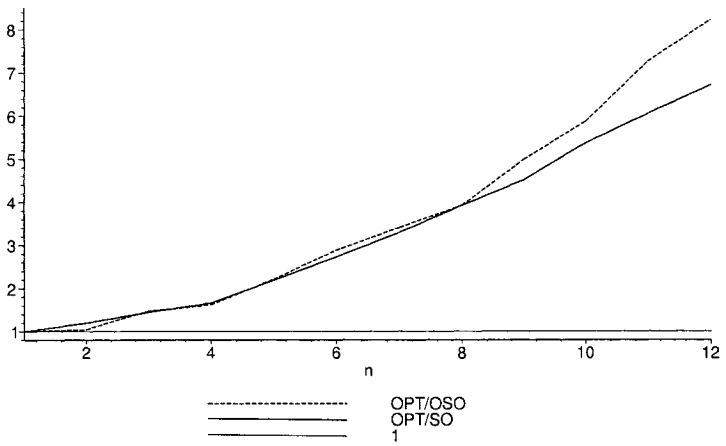
	Fonctions lissses			Fonctions C0			Fonctions LPM		
n	OPT	OSO	SO	OPT	OSO	SO	OPT	OSO	SO
1	.202647	.202647	.202647	.132599	.132599	.132599	.149931	.149931	.149931
2	.084956	.110451	.083263	.050911	.047532	.044185	.066827	.063943	.055701
3	.046675	.045253	.045200	.026635	.020802	.020343	.036388	.024703	.025253
4	.029604	.033644	.028013	.017157	.012088	.010664	.022481	.013871	.013541
5	.020394	.024208	.019103	.012242	.007131	.006765	.016440	.007423	.007506
6	.014921	.016445	.013838	.008554	.004609	.004417	.011757	.004087	.004310
7	.011370	.010649	.010510	.006349	.003162	.003154	.008927	.002616	.002705
8	.008952	.009134	.008263	.005024	.002331	.002326	.006674	.001700	.001700
9	.007227	.007835	.006612	.004136	.001778	.001791	.005759	.001157	.001277
10	.005959	.006673	.005384	.003255	.001389	.001385	.004617	.000787	.000860
11	.004998	.005603	.004488	.002872	.001109	.001074	.003968	.000547	.000656
12	.004252	.004666	.003829	.002459	.000899	.000865	.003238	.000394	.000482



(a) Fonctions lisses.



(b) Fonctions lisses par morceaux.



(c) Fonctions linéaires par morceaux.

Figure 7.4 Comparaison de OPT, OSO et SO pour l'approximation L_h^1 avec $h(t) = t(1-t)$.

7.6 Conclusion

Nos expériences montrent qu'il est pertinent d'examiner des algorithmes adaptatifs pour les problèmes d'approximation et ce même dans les cas où l'adaptation n'est pas utile. Connaissant un algorithme séquentiellement optimal idéal mais possiblement coûteux, on peut chercher des approximations qui seront presqu'aussi efficaces mais à moindre coût.

Pour l'approximation L^∞ la performance en dents de scie des algorithmes de type sandwich et OSO devrait être comprise pour pouvoir élaborer des algorithmes plus réguliers. Ce type d'algorithme est à notre avis très utile pour les problèmes d'approximation car ils sont non seulement efficaces et peu coûteux mais nous croyons qu'ils se généralisent aux problèmes de plus grandes dimensions.

Enfin, pour l'approximation pondérée d'autres tests seraient de mises pour tenter de comprendre l'effet de la fonction de poids h sur la performance des différents algorithmes.

CONCLUSION

Nous avons dans ce travail complété et unifié la théorie concernant l'aproximation des fonctions de la classe $F_{a,b}$. D'abord, des bornes inférieure et supérieure explicites pour l'approximation dans la norme L^p avec $p \geq 1$ nous ont permis de montrer que pour ces problèmes les algorithmes adaptatifs sont meilleurs dans le pire cas sauf si $p = 1$. Dans ce dernier cas, il existe un algorithme passif optimal. Ensuite, pour l'approximation dans la norme L^1 pondérée par une fonction de poids positive, nous avons montré qu'un algorithme adaptatif ne peut pas faire beaucoup mieux qu'un algorithme passif optimal.

Pour tous ces problèmes nous avons proposé des algorithmes d'ordre optimal et dans le cas L^1 et L^1_h un algorithme optimal et séquentiellement optimal. Ces différents algorithmes ont été comparés et nous avons constaté qu'en pratique l'efficacité de ces méthodes dépend du type de fonctions à approximer : lisses ou avec une dérivée discontinue. Il est donc utile d'avoir plusieurs méthodes à notre disposition de façon à pouvoir choisir celle qui présente le meilleur compromis entre efficacité et coût des calculs pour un problème donné.

Pour le difficile problème d'approximation dans la norme L^∞ nous avons proposé une formulation de programmation dynamique qui nous a permis de résoudre le problème pour $n = 1$ point d'évaluation et de le simplifier pour $n \geq 2$ points. Du travail reste encore à faire pour en tirer un algorithme optimal efficace mais nous croyons avoir fait un pas vers la solution de ce problème.

D'un point de vue général, ce travail montre que l'approche calculatoire que nous avons employée peut donner à la fois des résultats théoriques et pratiques. Les problèmes d'approximation tels ceux que nous avons étudiés sont intéressants parce qu'ils permettent de réfléchir sur les rapports entre la théorie et les applications pra-

tiques. Du point de vue théorique, on définit des limites à ce qui peut être accompli concrètement et il faut ensuite se poser la question de savoir quelles restrictions nous sont imposées par ces limites. Traub et Werschulz [44] vont plus loin dans ce sens et posent la question de savoir si les limites théoriques sont en fait des limites à la connaissance scientifique. Du point de vue des applications, l'étude des algorithmes optimaux donne des modèles pour construire des algorithmes efficaces.

Problèmes ouverts et généralisations

Certains problèmes abordés dans ce travail n'ont pas encore trouvé de solution complète. Nous les présentons avec un bref commentaire.

1. Les bornes inférieure et supérieure pour l'approximation L^p du chapitre 4 ne sont sans doute pas exactes. Est-il possible de déterminer des bornes exactes, particulièrement pour $p = \infty$? La difficulté ici est que l'on cherche à calculer une borne supérieure pour des algorithmes adaptatifs.
2. Pour l'approximation L^∞, peut-on construire un algorithme efficace à partir de la formulation de programmation dynamique (4.5) ?
3. Au chapitre 5 les bornes sur l'erreur pour l'approximation L_h^1 sont suffisantes pour montrer que l'adaptation n'est pas utile au sens large mais elles sont grossières. Peut-on en trouver de meilleures ?
4. Pour ce même problème, nous croyons qu'en fait l'adaptation n'est pas utile au sens stricte, c'est-à-dire que $e_{\text{ad}} = e_{\text{pas}}$. Peut-on démontrer ceci ? Nous croyons que le théorème (1.3) peut être utile ici.
5. Comment peut-on améliorer l'efficacité de la méthode de résolution de

$$\nabla G(x, v, \mu) = 0$$

pour l'approximation L_h^1 ? Nous croyons qu'une meilleure implantation pour-

rait déjà donner une procédure plus rapide.

La généralisation la plus immédiate de nos problèmes est de considérer des fonctions concaves de plus d'une variable. Les problèmes d'approximation en grande dimension dans la norme L^1 sont présentement des domaines de recherche très actifs et sont souvent étudiés dans le contexte stochastique. Dans ce cas les algorithmes aléatoires permettent souvent de vaincre la malédiction des grandes dimensions, en contrepartie d'une borne plus faible sur l'erreur moyenne de l'algorithme (voir par exemple Traub et Werschulz [44]). Le contexte du pire cas et les algorithmes déterministes sont toujours pertinents si on considère des fonctions avec peu de variables. Par exemple, une généralisation de l'exemple donné en introduction est le problème d'équilibre à trois critères qui peut survenir en pratique. Dans ce cas, on doit approximer des fonctions concaves de deux variables.

Sonnevend [34] a étudié le problème d'approximation L^∞ de fonctions concaves sur le carré $[0,1]^2$ et les algorithmes qu'il propose sont du type sandwich avec évaluations en plusieurs points à chaque itération. Pour l'approximation L^1 Sukharev [40] construit un algorithme séquentiellement optimal pour l'intégration itérée basé sur un algorithme d'approximation à une seule variable. Dans le même ordre d'idée Plaskota et Wasilkowski [29] utilisent eux aussi un algorithme optimal en une variable pour construire un algorithme d'intégration à plusieurs variables.

La plupart des auteurs considèrent les problèmes d'approximation de plus d'une variable pour des fonctions définies sur un cube $[0,1]^n$. Nous croyons qu'il est avantageux d'exploiter, comme nous l'avons fait pour une variable, la décomposition du problème d'approximation sur des «sous-intervalles»où sur chacun est reproduite la situation initiale lorsqu'aucune évaluation n'a été faite. Nous proposons donc de considérer des fonctions concaves sur le simplexe unité, qui est une généralisation naturelle de l'intervalle $[0,1]$ et qui se prête à cette décomposition. Ceci est illustré

à la figure 7.5.

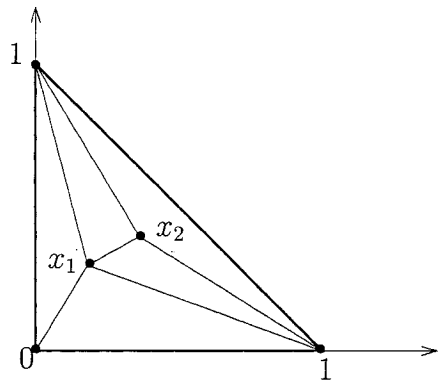


Figure 7.5 Décomposition du simplexe unité.

Les dérivées aux «extrémités»sont les gradients aux trois sommets de chaque sous-domaine. Avec cette décomposition on peut écrire une formulation de programmation dynamique du problème. Cependant l'utilisation de l'information donnée par les gradients est beaucoup plus difficile en deux variables et représente un défi de taille.

En conclusion, nous espérons avoir fait dans ce travail quelques pas vers la solution de différents problèmes d'approximation et que les directions proposées ci-dessus s'avéreront propices à de futures recherches.

RÉFÉRENCES

[1] BAKHVALOV, N. (1976). *Méthodes numériques*, Éditions Mir, Moscou, 1976.

[2] BAZARAA, S.M., SHERALI, H.D., SHETTY, C.M. (1993). *Nonlinear Programming : theory and algorithms, 2nd edition*, Wiley, New York, 1993.

[3] BELLMAN, R. and DREYFUS, R. (1962). *Applied Dynamic Programming*, Princeton University Press, Princeton, 1962.

[4] BURDEN, R.L., FAIRES, J.D. (2001). *Numerical Analysis, seventh edition*, Brooks/Cole, Pacific Grove, CA, 2001.

[5] BURKARD, R.E., HAMACHER, H.W., ROTE, G. (1991). Sandwich Approximation of Univariate Convex Functions with an Application to Separable Convex Programming, *Naval Research Logistics* **38**, 911-924.

[6] CERONE, P., ROUMELIOTIS, J., HANNA, G. (2000). On weighted three point quadrature rules, *ANZIAM Journal* **42**, C340–C361.

[7] CURBERA, F. (2000). Delayed Curse of Dimension for Gaussian Integration, *Journal of Complexity* **16**, 474-506.

[8] CHVÁTAL, V. (1983). *Linear Programming*, W.H. Freeman, New York, 1983.

[9] DEMYANOV, V.F, MALOZEMOV, V.N. (1974), *Introduction to Minimax*, Dover, New York, 1974.

[10] FORTIN, A. (1995), *Analyse numérique pour ingénieurs*, Éditions de l'École Polytechnique de Montréal, Montréal, 1995.

[11] GAL, S., MICCHELLI, A. (1980). Optimal sequential and non-sequential procedures for evaluating a functional, *Applicable Analysis* **10**, 105-120.

[12] GLINKIN, I.A. (1984). Best quadrature formula in the class of convex functions, *Mat. Zametki* **35**, 267-277.

[13] GROSS, O. (1956). A class of discrete type minimization problems, *Rand Corporation Research Memorandum no. 1644* (1956).

[14] GUÉRIN, J., MARCOTTE, P., SAVARD, G. (2001). An optimal adaptive algorithm for the approximation of concave functions, *Mathematical Programming*, à paraître.

[15] GUÉRIN, J. (2000). *Une méthode adaptative pour l'approximation de fonctions concaves croissantes*, Mémoire de maîtrise, École Polytechnique de Montréal.

[16] HICKERNELL, F.J., SLOAN, I.H., WASILKOWSKI, G.W. (2004). On tractability of weighted integration over bounded and unbounded regions in $\mathbf{R}^s$, *Mathematics of Computation* **73**, 1885-1901.

[17] KIEFER, J. (1957). Optimum sequential search and approximation methods under minimum regularity assumptions, *SIAM Journal* **5**, 105-136.

[18] KORNEICHUK, N. P. (1995). Information widths, *Ukrainian Mathematical Journal Vol.* **47** *no. 11*, 1720-1732.

[19] KORNEICHUK, N. P. (1999). Information aspects in the theory of approximation and recovery of operators, *Ukrainian Mathematical Journal Vol.* **51** *no. 3*, 353-365.

[20] LEE, M.-S. (2001). Approximation in weighted L_p spaces, *Numerical Functional Analysis and Optimization* **22** *nos. 5-6*, 657-674.

[21] MAPLE 10, *Logiciel de calcul symbolique et numérique*, Maplesoft, www.maplesoft.com (2005).

[22] MARCOTTE, P., NGUYEN, S., TANGUAY, K. (1996). Implementation of an efficient algorithm for the multiclass traffic assignement problem, *Proceedings of the 13th International Symposium on Transportation and Traffic Assignement Theory*, Lyon, Jean-Baptiste Lesort ed., Pergamon, 217-236.

[23] MATHÉ, P. (1998). Asymptotically optimal weighted numerical integration, *Journal of Complexity* **14**, 34-48.

[24] NOVAK, E. (1988). *Deterministic and Stochastic Error Bounds in Numerical Analysis*, Lecture Notes in Mathematics, Springer-Verlag, 1988.

[25] NOVAK, E. (1992). Quadrature formulas for monotone functions, *Proceedings of the American Mathematical Society Vol.* **115** *no. 1*, 59-68.

[26] NOVAK, E. (1993). Quadrature formulas for convex classes of functions, *International Series of Numerical Mathematics Vol.* **112**, Birkhäuser Verlag, Basel, 283-296.

[27] NOVAK, E. (1995). Optimal recovery and n-widths for convex classes of functions, *Journal of Approximation Theory* **80**, 390-408.

[28] NOVAK, E. (1996). On the power of adaption, *Journal of Complexity* **12**, 199-237.

[29] PLASKOTA, L., WASILKOWSKI, G.W. (2004). Smolyak's algorithm for integration and L_1-approximation of multivariate functions with bounded mixed derivatives of second order, *Numerical Algorithms* **36** *no.3*, 229-246.

[30] ROTE, G. (1992). The Convergence Rate of the Sandwich Algorithm for Approximating Convex Functions, *Computing* **48**, 337-361.

[31] SAATY, T.L. (1970). *Optimization in Integers and Related Extremal Problems*, McGraw-Hill, New York, 1970.

[32] SONNEVEND, G. (1978). On the optimization of adaptive numerical algorithms of approximation. *Operations Research Verfahren, vol.31*, Hain, Scriptor, Hanstein, Königstein eds., 581-595.

[33] SONNEVEND, G. (1983). Optimal passive and sequential algorithms for the approximation of convex functions in $L_p[0,1]^s$, $p = 1, \infty$, *Constructive function theory '81*, Sofia 1983.

[34] SONNEVEND, G. (1983). An optimal sequential algorithm for the uniform approximation of convex functions on $[0,1]^2$, *Applied Mathematics and Optimization* **10**, 127-142.

[35] SONNEVEND, G. (1984). Sequential algorithms of optimal order global error for the uniform recovery of functions with monotone $(r-1)$ derivatives, *Analysis Mathematica* **10**, 311-335.

[36] SUKHAREV, A. G. (1979). A sequentially optimal algorithm for numerical integration, *Journal of Optimization Theory and Aplications Vol.* **28**, *no. 3*, 363-373.

[37] SUKHAREV, A. G. (1986). On the existence of affine methods for approximating linear functionals, *Journal of Complexity* **2**, 317-322.

[38] SUKHAREV, A. G. (1987). The concept of sequential optimality for problems in numerical analysis, *Journal of Complexity* **3**, 347-357.

[39] SUKHAREV, A. G., CHUYAN, O.R. (1990). On adaptive and nonadaptive stochastic and deterministic algorithms, *Journal of Complexity* **6**, 119-127.

[40] SUKHAREV, A. G. (1992). *Minimax models in the theory of numerical methods*, Theory and Decision Library Series B Vol. **21**. Kluwer Academic Publishers, 1992.

[41] TRAUB, J. F., WOŹNIAKOWSKI, H. (1980). *A general theory of optimal algorithms*, Academic Press, New York, 1980.

[42] TRAUB, J. F., WASILKOWSKI, G. W., WOŹNIAKOWSKI, H. (1983). *Information, Uncertainty, Complexity*, Addison-Wesley, Reading, MA, 1983.

[43] TRAUB, J. F., WASILKOWSKI, G. W., WOŹNIAKOWSKI, H. (1988). *Information-Based Complexity*, Academic Press, New York, 1988.

[44] TRAUB, J. F., WERSCHULZ, A.G. (1999). *Complexity and Information*, Cambridge University Press, Cambridge, 1999.

[45] WASILKOWSKI, W.G., WOŹNIAKOWSKI, H. (2000). Complexity of weighted approximation over $\mathbf{R}^1$, *Journal of Complexity* **103**, 223-251.

[46] WOŹNIAKOWSKI, H. (2003). Open problems for tractability of multivariate integration, *J. Complexity* **19** *no.3*, 343-444.

[47] ZWICK, D. (1988). *Optimal Quadrature for Convex Functions and Generalizations*, International Series of Numerical Mathematics, **Vol 85**, Birkhäuser, Basel, 1988.

[48] WANG, X. (2003). Strong tractability of multivariate integration using quasi-Monte Carlo algorithms, *Math. Comp.* **72** *no. 242*, 823-838.

[49] YANG, X.Q., GOH, C.J. (1997). A method for convex curve approximation, *European Journal of Operational Research* **97**, 205-212.