

UNIVERSITÉ DE MONTRÉAL

COMPOSANTES PRINCIPALES MULTI-ÉCHELLES
AVEC SUPPORT COMPACT À DOUBLE CHEVAUCHEMENT

GUILLAUME NEPVEU
DÉPARTEMENT DE MATHÉMATIQUES ET DE GÉNIE INDUSTRIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLOME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(MATHÉMATIQUES APPLIQUÉES)
MAI 2008



Library and
Archives Canada

Bibliothèque et
Archives Canada

Published Heritage
Branch

Direction du
Patrimoine de l'édition

395 Wellington Street
Ottawa ON K1A 0N4
Canada

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

ISBN: 978-0-494-41572-6

Our file Notre référence

ISBN: 978-0-494-41572-6

NOTICE:

The author has granted a non-exclusive license allowing Library and Archives Canada to reproduce, publish, archive, preserve, conserve, communicate to the public by telecommunication or on the Internet, loan, distribute and sell theses worldwide, for commercial or non-commercial purposes, in microform, paper, electronic and/or any other formats.

The author retains copyright ownership and moral rights in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

AVIS:

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, publier, archiver, sauvegarder, conserver, transmettre au public par télécommunication ou par l'Internet, prêter, distribuer et vendre des thèses partout dans le monde, à des fins commerciales ou autres, sur support microforme, papier, électronique et/ou autres formats.

L'auteur conserve la propriété du droit d'auteur et des droits moraux qui protègent cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé:

COMPOSANTES PRINCIPALES MULTI-ÉCHELLES
AVEC SUPPORT COMPACT À DOUBLE CHEVAUCHEMENT

présenté par: NEPVEU Guillaume

en vue de l'obtention du diplôme de: Maîtrise ès sciences appliquées

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

M. SOUMIS François, Ph.D., président

M. SAUCIER Antoine, Ph.D., membre et directeur de recherche

M. BOYER François-Raymond, Ph.D., membre

REMERCIEMENTS

Je tiens à remercier tout d'abord mon directeur de recherche M. Antoine Saucier pour m'avoir proposé à la fin de mon baccalauréat un projet de fin d'études qui me stimula grandement à poursuivre des études supérieures à l'École Polytechnique de Montréal. Je veux aussi le remercier pour son soutien tout au long de ma maîtrise, intellectuellement et financièrement. Je tiens à le féliciter pour sa disponibilité, son encadrement, son savoir et ses talents de vulgarisateur.

Je tiens aussi à remercier M. François Soumis et M. Charles Audet pour l'aide précieuse qu'ils ont apportée dans la résolution du problème d'optimisation étudié.

Merci à Mme. Carole Burney-Vincent et André Dupont pour m'avoir accordé les postes de chargé de cours et de travaux dirigés qui me procurèrent une stabilité financière propice à la réalisation de mes travaux. Merci aussi à M. Jean Guérin pour m'avoir permis, via le centre de consultation en mathématiques, d'aider les étudiants au baccalauréat et de parfaire mes connaissances en mathématiques.

Je voudrais remercier l'École Polytechnique de Montréal pour m'avoir fourni un climat académique adéquat permettant la réalisation de mon projet de maîtrise.

Je tiens finalement à remercier ma famille et mes amis pour m'avoir remonté le moral lors des périodes difficiles. Merci à Simon de Montigny pour son aide en Latex.

RÉSUMÉ

Dans ce projet, on propose une modification aux composantes principales multi-échelles (MPC). Les MPC sont des bases orthonormales adaptatives multi-échelles avec support compact. L'objectif du projet est d'augmenter l'adaptabilité de ces bases multi-échelles tout en préservant la proportion de vecteurs de petites échelles par l'introduction d'un double chevauchement entre les supports des vecteurs sur chaque niveau de petites échelles. Ces nouvelles bases portent le nom de composantes principales multi-échelles à double chevauchement (DOMPC). À partir du problème d'optimisation des MPC disjointes, ce chevauchement se traduit par l'ajout de deux nouvelles contraintes quadratiques d'orthogonalité. Deux méthodes ont été explorées pour résoudre ce nouveau problème d'optimisation.

La première méthode appelée le *Zipper* est basée sur un algorithme itératif qui consiste à générer des vecteurs qui convergent vers une solution du problème d'optimisation. Partant du fait que chaque vecteur d'un même niveau de petite échelle doit être orthogonal à deux décalages de lui-même, on génère successivement un vecteur qui répond au problème d'optimisation et qui est orthogonal à des décalages de deux vecteurs générés lors des itérations précédentes. À la première itération, on choisit des vecteurs initiaux aléatoires. Cette approche permet à première vue de résoudre le problème d'optimisation sans avoir à y ajouter les deux contraintes quadratiques d'orthogonalité qui le rendent plus complexe à solutionner. Malgré la simplicité de cette méthode, les résultats obtenus manquent de fiabilité. En effet la convergence de cet algorithme n'est pas garantie. Lorsque la convergence n'est pas obtenue, on obtient des solutions qui ne respectent pas les contraintes quadratiques. Après plusieurs tentatives infructueuses pour pallier à ce problème, cette approche a été abandonnée.

La seconde approche consiste à résoudre le problème d'optimisation en approximant la solution en effectuant la recherche d'une solution dans un sous-espace linéaire. On propose de chercher la solution sous la forme d'une combinaison linéaire de trois vecteurs propres ayant des valeurs propres faibles. Ces vecteurs propres sont obtenus à partir de la matrice de corrélation du signal calculée à chacun des niveaux des petites échelles. Cette approche permet d'obtenir une solution approximative du problème d'optimisation et de trouver des solutions qui satisfont les contraintes quadratiques contrairement à la première approche. Les bases DOMPC obtenues ont la même proportion de vecteurs de petites échelles que les bases MPC, mais les vecteurs des bases DOMPC possèdent environ trois fois plus de degrés de liberté. Cela représente une amélioration significative de l'adaptabilité des bases générées.

On a procédé à des expériences de compression et de filtrage pour comparer les performances des bases DOMPC par rapport à certaines bases d'ondelettes. Huit signaux ont été étudiés : une sinusoïde, une onde carrée, une onde triangulaire, une onde oscillante de longue période, un bruit blanc lissé, une section 1D d'une image d'empreinte digitale, un extrait de musique numérisé et un signal constant par morceaux. La compression a été effectuée avec les bases DOMPC-9, DOMPC-15, Daubechies-4 et Daubechies-8.

Les bases DOMPC performant beaucoup mieux que les ondelettes pour les signaux de faible dimension (sinusoïde, onde carrée, onde triangulaire et onde oscillante de longue période). Par exemple les bases DOMPC-9 et DOMPC-15 reproduisent la sinusoïde presque parfaitement ($\text{SNR} > 250 \text{ dB}$) avec seulement 1% et 7% des coefficients, comparativement aux bases d'ondelettes qui nécessite environ 50% des coefficients pour reproduire une reconstruction de 100 dB. Des résultats similaires ont été obtenus pour l'onde carrée, l'onde triangulaire et le signal oscillant de longue période.

Pour les autres signaux (bruit blanc lissé, empreinte digitale, extrait de musique et signal constant par morceaux), les performances de compression pour les bases DOMPC et d'ondelettes sont plus semblables. Les bases DOMPC performant aussi bien que les ondelettes pour le bruit blanc lissé et pour l'image d'empreinte digitale. Pour l'extrait de musique, les bases DOMPC performant mieux que les bases d'ondelettes avec un SNR supérieur de 10 à 15 dB, ce qui correspond environ à un facteur dix. Pour le signal constant par morceaux, les ondelettes performant mieux que les bases DOMPC.

Les expériences de filtrage ont été effectuées sur la sinusoïde, l'onde carrée et le signal constant par morceaux. Le bruit ajouté est un bruit blanc uniformément distribué multiplié par une fonction gaussienne centrée. Le filtrage a été effectué avec les bases DOMPC-15, Daubechie-8, Coiflet-3 et Symlet-8. Deux types de filtrage ont été testés : le *hard thresholding* et le *shrinking*. Les résultats montrent que la base DOMPC est plus efficace pour filtrer la sinusoïde et l'onde carrée que les bases d'ondelettes. Par exemple, pour le filtrage de type *hard thresholding* de la sinusoïde, la base DOMPC permet d'obtenir un SNR d'environ 47 dB contre 23 dB pour la meilleure base d'ondelettes. Les performances pour l'onde carrée sont semblables à celles obtenues pour de la sinusoïde. Pour le signal constant par morceaux, les performances de la base DOMPC sont comparables à celles des bases d'ondelettes fournissant des SNR variants de 10 dB à 12 dB.

ABSTRACT

In this project, we propose an improvement to multiscale principal components (MPC). MPC are orthonormal adaptive multiscale bases with compact support. The objective of this project is to increase the adaptability of these multiscale bases while preserving the proportion of small scale vectors by adding a double overlap between the vectors' supports on each small scale levels. These new bases are called double overlap multiscale principal components (DOMPC). Starting from the MPC's optimization problem, this overlap introduces two new quadratic orthogonality constraints. Two methods for resolving this new optimization problem have been explored.

The first method called the *Zipper* is an iterative method generating vectors that converge toward an optimization problem's solution. Each small scale vector must be orthogonal to two translations of itself. We generate vectors that satisfy the optimization problem and that are orthogonal to shifted vectors generated at previous iterations. At the first iteration, we choose two initial random vectors. This approach seems to solve the optimization problem without adding to it the two orthogonal quadratic constraints, which makes it more complex to solve. Despite this method's simplicity, it lacks reliability. In fact, the algorithm's convergence is not guaranteed. When there is no convergence, we obtain solutions that do not respect the orthogonal quadratic constraints. After many unsuccessful attempts to settle this problem, this approach was abandoned.

The second method consists of searching an approximate optimization problem's solution by search in vector subspace. We propose to search a solution in form of a linear combination of three eigenvectors associated to small eigenvalues. Those eigenvectors are obtained from the signal's correlation matrix calculated at each small scale levels. This approach gives an approximate optimization problem's solution that respects the quadratic orthogonal constraints, contrarily to the first method. The DOMPC bases obtained have the same proportion of

small scale vectors than MPC bases, but the bases vectors have about three times the number of degree of freedom. This represents a significant improvement of the bases' adaptability.

We proceeded to compression and denoising experiments to compare the performances of the DOMPC bases and some wavelet bases. A total of eight signals were studied: a sinusoidal signal, a square wave, a triangular wave, a long-period oscillating signal, a smoothed noise, an one-dimensional section of a fingerprint image, an instrumental music signal and a piecewise constant signal. The compression was done using DOMPC-9, DOMPC-15, Daubechies-4 and Daubechies-8 bases.

The DOMPC bases performed much better than the wavelets for low-dimensional signal (sinusoidal signal, square wave, triangular wave and long-period oscillating signal). For instance, DOMPC-9 and DOMPC-15 bases provide a near perfect reconstruction ($\text{SNR} > 250 \text{ dB}$) of the sinusoidal signal with only 1% and 7% of the total coefficients, compared to the wavelet bases that need about 50% of the coefficients for a reconstruction of 100 dB. Similar results are obtained for the square wave, triangular wave and the long-period oscillating signal.

For the other signals (smoothed noise, one-dimensional section of a fingerprint image, instrumental music signal and piecewise constant signal), the compression performances of the DOMPC and wavelet bases are more similar to each other. The DOMPC bases perform as well as wavelet bases for the smoothed noise and fingerprint image. For the instrumental music signal, DOMPC bases perform better than wavelets providing an SNR higher by 10 to 15 dB, which corresponds roughly to a factor of ten. For the piecewise constant signal, the wavelet bases perform better than the DOMPC bases.

The denoising experiments were done on the sinusoidal signal, the square wave and the piecewise constant signal. The added noise is a uniformly distributed zero-mean and unit variance white noise which is multiplied by a centered gaussian. We used the DOMPC-15, Daubechie-8, Coiflet-3 and Symlet-8 bases. Two different types of filtering were used: *hard thresholding* and *shrinking*. The results show that the DOMPC basis performs much better than the wavelets for denoising the sinusoidal signal and the square wave. For instance, using *hard thresholding* filtering on the sinusoidal signal, the DOMPC basis provides a SNR of 47 dB compared to 23 dB for the best wavelet. The performances of the DOMPC basis for the square wave signal are comparable to those obtained with the sinusoidal signal. For the piecewise constant signal, the performances of the DOMPC basis and the wavelet bases are similar providing SNR from 10 dB to 12 dB.

TABLE DES MATIÈRES

REMERCIEMENTS.....	iv
RÉSUMÉ	v
ABSTRACT.....	viii
TABLE DES MATIÈRES.....	xi
LISTE DES FIGURES	xiii
LISTE DES ABRÉVIATIONS	xiv
LISTE DES ANNEXES	xv
CHAPITRE 1 INTRODUCTION.....	1
CHAPITRE 2 REVUE CRITIQUE DE LITTÉRATURE	5
CHAPITRE 3 ORGANISATION DU TRAVAIL	8
CHAPITRE 4 MULTISCALE PRINCIPAL COMPONENTS WITH DOUBLE OVERLAP COMPACT SUPPORT	9
4.1 Introduction.....	10
4.2 Definition and properties of double overlap MPC.....	14
4.2.1 Double overlap structure of the vector supports.....	14
4.2.2 Number of degrees of freedom	15
4.2.3 Proportion of small scales vectors	16
4.2.4 Comparison of $N_{DF}(n)$ for disjoint MPC and DOMPC.....	18
4.3 Construction of double-overlap MPC.....	19

4.3.1 Correlation Matrix	19
4.3.2 Formulation of the optimization problem	20
4.3.3 Solution of the optimization problem	20
4.3.4 Examples of DOMPC	25
4.4 Description of reference signals	25
4.5 Data compression experiments	30
4.5.1 Data compression method	30
4.5.2 Data compression results	31
4.5.3 Interpretation	32
4.6 Noise filtering experiments	34
4.6.1 Noise filtering method	34
4.6.2 Noise filtering results	40
4.7 Conclusion	43
Bibliography	45
CHAPITRE 5 MÉTHODE DU « ZIPPER »	47
CHAPITRE 6 SYNTHÈSE et CONCLUSION	50
6.1 Discussion	50
6.2 Conclusion et recommandations	51
RÉFÉRENCES BIBLIOGRAPHIQUES	52
ANNEXE	56

LISTE DES FIGURES

Figure 4.1	Support locations for a DOMPC basis	14
Figure 4.2	Small scale vectors for a DOMPC basis	26
Figure 4.3	Samples of the reference signals used in this paper	28
Figure 4.4	A fingerprint image	29
Figure 4.5	SNR versus $100 N_C/N$ for a) the sinusoid; b) the square wave; c) the triangle wave; d) the long-period oscillating signal.....	34
Figure 4.6	SNR versus $100 N_C/N$ for a) the smoothed white noise; b) the fingerprint signal; c) the music recording; d) the piecewise constant signal.....	35
Figure 4.7	SNR versus $100 N_C/N$ for the sinusoidal signal	36
Figure 4.8	SNR versus $100 N_C/N$ for the piecewise constant signal.....	37
Figure 4.9	Example of the sinusoidal signal (a), the localized synthetic noise (b) and the sum (c).....	39
Figure 4.10	Filtering residue $E = S - S_F$ for the noisy sinusoidal signal using hard thresholding (left column) and shrinking thresholding (right column)	41
Figure 4.11	Filtering residue $E = S - S_F$ for the noisy square wave signal using hard thresholding (left column) and shrinking thresholding (right column)	42
Figure 5.1	Position des supports des vecteurs des petites échelles sur un même niveau (A) et substitutions à effectuer à chaque itération pour la méthode itérative (B).....	48

LISTE DES ABRÉVIATIONS

DOMPC	:	Double Overlap Multiscale Principal Components
MPC	:	Multiscale Principal Components
PCA	:	Principal Components Analysis
SNR	:	Signal-to-Noise ratio

LISTE DES ANNEXES

ANNEXE A : CODE *MATLAB* DE LA MÉTHODE DE RÉOLUTION
PAR APPROXIMATION.

CHAPITRE 1 INTRODUCTION

Dans le monde d'aujourd'hui, on utilise une abondance de signaux numériques notamment à cause de l'informatisation et de la croissance des technologies de l'information. Un signal numérique peut être décrit comme une information représentée sous la forme d'une suite ordonnée de nombres. Plusieurs domaines utilisent les signaux numériques comme par exemple la téléphonie cellulaire, le transfert et le stockage de musiques sur ordinateur (e.g. le format de musique mp3), les réseaux de diffusion télévisuels, etc. Le domaine du traitement de signal est une discipline qui étudie et développe des algorithmes d'analyse et d'interprétation de signaux. On y utilise notamment des représentations des signaux dans diverses bases vectorielles. Une base qui exprime un signal avec un maximum de coefficients nuls ou quasi-nuls est un exemple d'une bonne représentation.

La transformée de Fourier discrète est un exemple de représentation utilisée en traitement de signal. Cette représentation décompose le signal analysé dans une fenêtre en une somme de sinus et de cosinus de diverses fréquences. Cette transformée permet d'obtenir le spectre fréquentiel d'un signal. Comme les supports des vecteurs de base (sinus et cosinus) couvrent la totalité de la taille de la fenêtre d'analyse, la transformée de Fourier est mal adaptée à un traitement ou une analyse locale du signal, e.g. le filtrage d'un bruit localisé.

L'analyse en ondelettes discrètes est une autre représentation utilisée en traitement de signal. La transformée en ondelettes discrètes permet de projeter un signal S dans une base orthonormale multi-échelles. Si on dénote les vecteurs de base par $\Psi_{n,m}$, où n est un paramètre d'échelle et m un paramètre de localisation des vecteurs, alors on peut exprimer le signal sous la forme

$$\mathbf{S} = \sum_{n=1}^N \sum_{m=1}^{M(n)} (\boldsymbol{\Psi}_{n,m}' \mathbf{S}) \boldsymbol{\Psi}_{n,m}. \quad (1.1)$$

Les vecteurs de niveau $n < N$ sont des vecteurs de petites échelles. Ces vecteurs sont non-nuls sur un intervalle restreint et nuls (ou quasi-nuls) ailleurs. Les vecteurs de niveau $n = N$ sont des vecteurs de grande échelle, qui ont un support qui couvre la totalité de la fenêtre. La transformée en ondelettes offre la possibilité d'utiliser une multitude de bases d'ondelettes différentes ce qui permet une certaine latitude lors de l'analyse d'un signal. Cependant, ce vaste nombre d'ondelettes peut aussi causer un problème de choix de la base optimale pour analyser un signal. Les bases d'ondelettes ont été créées de manière à ce que leurs vecteurs soient orthogonaux aux polynômes de degré inférieur à k . Ce dernier paramètre dépend de l'ondelette utilisée. La décomposition d'un signal polynomial par morceaux produira ainsi un grand nombre de coefficients d'ondelettes nuls ou quasi-nuls.

L'analyse en composantes principales, aussi connue sous le nom de transformée de Karhunen-Loève, est une autre représentation couramment utilisée en traitement de signal. Les composantes principales permettent de représenter un signal $\mathbf{S}$ dans une base orthonormale $\{\mathbf{V}_i, i=1, \dots, P\}$ choisie de manière à ce que l'erreur

$$\left\| \mathbf{S} - \sum_{j=1}^p (\mathbf{V}_j' \mathbf{S}) \mathbf{V}_j \right\|, \quad (1.2)$$

basée sur la norme euclidienne, soit minimale pour $p=1, \dots, P$. On peut montrer que cette base vectorielle est composée des vecteurs propres orthonormaux de la matrice de corrélation du signal analysé. Ces vecteurs sont classés en ordre décroissant par rapport aux valeurs propres. L'approximation d'un signal est obtenue en ne conservant que les coefficients associés aux

premiers vecteurs de base. Il s'agit d'une représentation qui est adaptative au signal analysé, c'est-à-dire que les vecteurs de base sont créés en fonction de ce signal. Tout comme la transformée de Fourier, l'analyse en composante principale est mal adaptée à l'analyse locale d'un signal. En effet, la taille des supports des vecteurs couvre la totalité de la fenêtre.

Les excellents résultats obtenus par l'application de la transformée en ondelettes d'une part et de l'analyse en composantes principales d'autre part laissent penser qu'il serait intéressant d'explorer une nouvelle méthode qui combinerait la capacité de réduction de la dimension de l'analyse en composantes principales et la flexibilité de l'analyse multi-échelle de la transformée en ondelettes. C'est dans cette optique que A. Saucier a développé les composantes principales multi-échelles disjointes (MPC). Cette nouvelle transformée est une généralisation de l'analyse en composantes principales. Tout comme la transformée en ondelettes, les MPC possèdent des vecteurs de petite et de grande échelles. Sur un même niveau de petite échelle, les vecteurs sont tous identiques à un certain nombre de décalages près. Les vecteurs de base sont calculés de manière à minimiser en moyenne le carré de leur produit scalaire avec le signal. Les vecteurs des petites échelles sont en fait des composantes principales calculées à différentes résolutions (facteurs d'échelle). Les vecteurs de grande échelle ont un support qui couvre toute la fenêtre.

Les vecteurs des bases MPC sont générés à partir d'un problème d'optimisation visant créer des bases adaptées au signal analysé. L'adaptabilité des bases MPC est contrôlée directement par le nombre de degrés de liberté de ses vecteurs de base. À un niveau donné, le nombre de degrés de liberté d'un vecteur est calculé en effectuant la soustraction entre la taille (ou diamètre) du support du vecteur et le nombre de contraintes qu'il doit satisfaire. La capacité d'analyse locale d'une base est liée à la proportion de vecteurs de petites échelles par rapport au nombre de vecteurs total.

Ce projet vise à modifier la structure des bases MPC disjointes afin d'en augmenter l'adaptabilité tout en conservant leur capacité d'analyse locale. Pour les MPC disjointes, les supports des vecteurs du premier niveau doivent avoir un diamètre L_0 d'au moins trois points pour obtenir une croissance du nombre de degrés de liberté d'un niveau à l'autre. Pour $L_0 = 3$, la base a une proportion de vecteurs de petites échelles d'environ $2/3$.

Si on veut augmenter le nombre de degrés de liberté des bases MPC, il faut augmenter le diamètre des supports des vecteurs. Cependant, cette augmentation entraîne une diminution de la proportion de vecteurs de petites échelles et donc une réduction de la capacité d'analyse locale de la base. Pour compenser cette réduction, on propose d'ajouter un chevauchement dans la structure des supports des vecteurs ce qui permettra d'augmenter la taille des supports tout en maintenant la proportion de vecteurs de petites échelles. Notre objectif est donc d'augmenter l'adaptabilité des bases MPC disjointes tout en conservant leur capacité d'analyse locale. Le cas de l'ajout d'un simple chevauchement a été développé parallèlement à ce projet. On étudie dans ce mémoire l'introduction d'un double chevauchement. Les bases ainsi créées portent le nom de composantes principales multi-échelles à double chevauchement (DOMPC). Ce double chevauchement modifie le problème d'optimisation utilisé pour générer les vecteurs de base. Deux contraintes quadratiques d'orthogonalité doivent être ajoutées afin que les vecteurs de petites échelles soient orthogonaux. Afin de tester l'efficacité de bases DOMPC, des expériences de compression et de filtrage sont effectuées pour comparer les performances des DOMPC et des ondelettes.

CHAPITRE 2 REVUE CRITIQUE DE LITTÉRATURE

Cette section fait un bref historique des domaines reliés au sujet de ce mémoire en traitement de signal. On y expose tout d'abord quelques articles portant sur la transformée en ondelettes, sur les paquets d'ondelettes et sur l'analyse en composantes principales. On présente ensuite quelques articles sur des méthodes utilisant à la fois la transformée en ondelettes et l'analyse en composantes principales.

La plupart des applications de la transformée en ondelettes discrètes reposent sur sa capacité d'approximer efficacement certaines familles de signaux avec un nombre restreint de coefficients non-nuls [13]. En effet, les ondelettes ont été bâties de manière à être orthogonales aux polynômes. Par exemple les ondelettes de type Daubechies-L [3] possèdent des moments d'ordre $\leq L$ nuls, ce qui les rend orthogonales aux polynômes de degré $L-1$. La transformée en ondelettes a été largement utilisée en compression [19, 20] et en filtrage [4, 5, 17, 20] de signaux. Notamment, les ondelettes sont utilisées dans le nouveau format d'image JPEG2000 qui remplacera l'ancien standard JPEG [21]. Pour contrôler l'amplitude des coefficients d'ondelettes, plusieurs méthodes ont été développées. Une approche a été développée pour contrôler l'amplitude des coefficients d'ondelettes à partir des moments nuls par Geronimo et *al.* [7]. D'autres méthodes incluent le *matching pursuit algorithm* proposé par Mallant et Zang [12], l'approche spectrale de Lilly et Park [11] et les paquets d'ondelettes [10]. Yiou et *al.* [22] proposa de construire des fonctions adaptatives basées sur les composantes principales.

La transformée en ondelettes discrètes est effectuée en appliquant itérativement un filtre passe-bas sur le signal analysé. À chaque itération, on calcule les coefficients de petites échelles à l'aide d'un filtre passe-haut. Le fait d'itérer uniquement avec le filtre passe-bas suppose que les informations importantes du signal sont contenues dans les basses fréquences. Pour plusieurs signaux, cette supposition n'est pas justifiée. À chaque itération, il existe la possibilité d'appliquer le filtre passe-bas, le filtre passe-haut ou même aucun filtre. L'analyse par paquets d'ondelettes utilise cette souplesse au niveau du choix des filtres à utiliser (ou non) à chaque itération [20]. L'analyse par paquets d'ondelettes a été utilisée dans la compression d'images [14, 16]. Afin de maximiser les performances en compression, un codage des coefficients est nécessaire.

L'analyse en composantes principales [8, 15] a été largement utilisée en traitement de signal à cause de sa capacité de fournir des approximations fidèles au signal avec un nombre limité de coefficients.

Certaines méthodes utilisant à la fois les ondelettes et les composantes principales ont été développées. Elles consistent à effectuer une décomposition en composantes principales sur les coefficients d'une transformée en ondelettes. Il y a notamment B.R. Bakshi qui proposa d'utiliser les *multiscale* PCA (MSPCA) dans le domaine du contrôle de processus statistiques à plusieurs variables [2]. Les auteurs Z. Geng et Q. Zhu ont par la suite raffiné la méthode de B.R. Bakshi et ont procédé à des expériences de filtrage [6]. M. Aminghafari, N. Cheze et J.-M. Poggi ont aussi proposé d'utiliser les MSPCA pour des applications de filtrage [1]. Les MSPCA ont aussi été utilisées pour la séparation des rythmes cardiaques maternel et foetal par E. C. Karvounis et D. I. Fotiadis [9].

Une nouvelle méthode a été proposée comme généralisation du théorème des composantes principales par A. Saucier [18]. Elle consiste à former des bases orthonormales multi-échelles adaptatives nommées composantes principales multi-échelles. Les MPC sont des représentations nouvelles qu'il ne faut pas confondre avec les MSPCA développés par B.R. Bakshi [2].

CHAPITRE 3 ORGANISATION DU TRAVAIL

Dans ce projet, notre objectif est d'augmenter l'adaptabilité des bases MPC [18] tout en conservant une même proportion de vecteurs de petites échelles. Afin d'atteindre cet objectif, on propose d'introduire un double chevauchement dans les supports des vecteurs d'un même niveau. Ce chevauchement ajoute deux contraintes quadratiques d'orthogonalité dans le problème d'optimisation des MPC disjointes. Afin de résoudre ce problème, une première méthode itérative a été testée et est décrite au chapitre 5. Cependant, étant donné que cette méthode ne converge pas tout le temps, cette première approche a été écartée. Une deuxième méthode basée sur l'approximation de la solution dans un sous-espace tridimensionnel est décrite dans l'article au chapitre 4. Dans cet article intitulé *Multiscale Principal Components with double overlap compact support*, la méthode complète pour produire les bases DOMPC est décrite et des résultats d'expériences de compression et de filtrage y sont présentés. On compare les bases DOMPC et certaines bases d'ondelettes. L'article du chapitre 4 est identique à celui soumis à la revue *Signal Processing*. Le chapitre 6 résume l'ensemble des résultats obtenus et on y propose des recommandations sur le projet.

Chapitre 4

MULTISCALE PRINCIPAL COMPONENTS WITH DOUBLE OVERLAP COMPACT SUPPORT

Guillaume Nepveu

*École Polytechnique de Montréal, Department of Mathematics and Industrial
Engineering, C.P. 6079, succ. centre-ville, Montreal (Quebec), Canada, H3C*

3A7. tel: (1) 514-340-4711 # 4516. Fax: (1) 514-340-4463. e-mail:

guillaume.nepveu@polymtl.ca.

Antoine Saucier

*Corresponding author. École Polytechnique de Montréal, Department of
Mathematics and Industrial Engineering, C.P. 6079, succ. centre-ville,
Montreal (Quebec), Canada, H3C 3A7. tel: (1) 514-340-4711 # 4516. Fax:*

(1) 514-340-4463. e-mail: antoine.saucier@polymtl.ca.

Abstract

Multiscale principal components (MPC) are a data adaptive alternative to discrete wavelet bases. MPC are designed to minimize the magnitude of the transform coefficients for a given class of signals. We introduce double overlap MPC (DOMPC) as an improvement of disjoint MPC. We show that DOMPC bases achieve a significantly higher degree of data-adaptability than disjoint MPC, while preserving a high proportion of small scale vectors (PSSV). DOMPC are the solution of an optimization problem involving quadratic constraints. One of the main contributions of this paper is an approximate solution method for this problem. We compare the performance of DOMPC and wavelets in data compression and noise filtering experiments. For low-dimensional signals (sinusoid, triangular or square waves, long-period oscillatory signals), we found that DOMPC perform much better than wavelets. In contrast with wavelets, DOMPC can perform very well for non-smooth signals. For a digital recording of instrumental music, we found that DOMPC perform significantly better than wavelets in data compression. For one-dimensional cuts through a fingerprint image, DOMPC and wavelets have comparable performances in compression. For piecewise constant signals, wavelets perform better than DOMPC in data compression because DOMPC bases have a lower PSSV than wavelets.

Keywords: Multiscale principal components, principal component analysis, wavelets, data-adaptive multiscale bases, optimization, data-compression, noise filtering.

4.1 Introduction

Most applications of wavelet bases exploit their ability to approximate efficiently specific families of functions with few non-zero wavelet coefficients (Mallat [10]-

b). These coefficients are scalar products $\psi^T \mathbf{S}$ of a wavelet ψ with the signal $\mathbf{S}$. If $\mathbf{S}$ is smooth and ψ has enough vanishing moments, then the wavelet coefficients are small at small scale. To control the magnitude of wavelet coefficients, several methods have been developed. One approach is to control the coefficients magnitude via vanishing moments, e.g. Geronimo *et al.* [5]. Other methods include the matching pursuit algorithm (Mallat and Zhang [9]), the spectral approach of Lilly and Park [8] and wavelet packets (Learned and Will-sky [7]). Yiou *et al.* [13] proposed to construct data-adaptive functions based on principal components to analyse data.

Data-adaptive multiscale bases taking advantage of existing wavelets bases have been developed by Bakshi [2], who applied principal component analysis (PCA) to the wavelet transform coefficients in the context of multivariate statistical process monitoring. This method, which is known as multiscale PCA, has been used for noise filtering purposes [4] [1] and for the maternal and fetal heart rate extraction from abdominal recordings [6].

Principal component analysis (PCA) and wavelets have both been used in signal processing for data compression [11] and noise filtering [3]. On one hand, principal component analysis (also known as Karhunen-Love expansions) consists in representing the signal $\mathbf{S}$ in a suitably chosen orthonormal basis $\{\mathbf{V}_i, i = 1, \dots, N\}$ for which the approximation error $\|\mathbf{S} - \sum_{i=1}^k (\mathbf{V}_i^T \mathbf{S}) \mathbf{V}_i\|$, based on the euclidian norm, is minimal for $k = 1, 2, \dots, N$. For $k < N$, the approximation $\mathbf{S} \approx \sum_{i=1}^k (\mathbf{V}_i^T \mathbf{S}) \mathbf{V}_i$ corresponds to a signal representation in a lower dimensional space. On the other hand, a wavelet basis can also be used to obtain a compressed representation of a signal. Compression is possible if many signal components vanish, which happens if the signal is piecewise smooth. One of the main differences between PCA and wavelets is that wavelets form multiscale bases, whereas principal components are not subject to localization

constraints.

Multiscale principal components (MPC) were introduced by Saucier [12] as a multiscale generalization of principal component analysis, as well as a data adaptive alternative to discrete wavelet bases. This approach consists in constructing a multiscale basis of vectors which is optimized so that the vector components are as close to zero as possible for a given class of signals. The resulting multiscale vectors are called *Multiscale Principal Components* (MPC), not to be confused with the multiscale PCA introduced by Bakshi [2].

An MPC basis is constructed in a finite size window which contains a sampled signal $\mathbf{S} \in \mathbb{R}^N$. Each vector of the basis has a compact support. The basis is composed of small scale and large scale vectors. Small scale vectors have a support size smaller than the window size, whereas large scale vectors have a support size which is equal to the window size. The support size of a level n vector is $L_n = 2^{n-1}L_0$, $n = 1, 2, \dots, n_{\max}$, where $L_0 > 0$ is an integer and $L_{n_{\max}} = N$. The smallest support diameter is therefore L_0 , which correspond to level one vectors. At any given construction level, all small scale vectors are constructed by translation of a single vector, i.e. the vectors are identical up to a translation shift.

Multiscale principal components are designed to minimize the coordinates mean square value for a given class of signals. The minimization problem is not tackled globally, but rather level by level, starting from the smallest scale vectors. Level n basis vectors are constrained to be orthogonal to all lower level vectors. Under this constraint, they are optimized to minimize their coordinates mean square value. The procedure is pursued toward large scales until level $n_{\max}$ is reached, where $n_{\max}$ is an adjustable parameter. At level $n_{\max}$, we finally construct a set of complementary large scale vectors, all of size $L_{n_{\max}} = N$, until a complete orthonormal basis of $\mathbb{R}^N$ is obtained. Large scale vectors are

built one by one so that the i^{th} large scale vector minimizes its coordinate mean square value while being constrained to be orthogonal to all previously constructed vectors.

In this paper, our main objective is to design MPC bases for which basis vectors have an improved degree of data adaptivity with respect to disjoint MPC [12]. An improved adaptivity leads to a larger percentage of negligible coefficients, which increases performance for data compression and noise filtering. For an MPC basis, improving the degree of data adaptivity can be achieved by increasing the number of degrees of freedom available for the optimization. For disjoint MPC, an increase of the vector support size leads to a corresponding increase of the number of degrees of freedom, but also reduces the proportion of small scale vectors (PSSV) in the basis, which may be a drawback for some applications. To compensate for this reduction, we propose in this paper to examine vector bases for which the support of each level- n basis vector overlaps with two other vectors at the same level. The resulting bases will be called *double-overlap multiscale principal components* (DOMPC). Our other objective in this paper is to compare the performance of DOMPC and wavelets in data compression and noise filtering experiments.

Notations and terminology: In this paper, we focus on one dimensional real signals defined at discrete coordinates $t_i \in \mathbb{R}$, $i = 1, 2, \dots, N$. The value of a signal or a function f at $t = t_i$ is denoted by $f(i)$. The signal is regarded as a column vector defined by $\mathbf{f} = (f(1), f(2), \dots, f(N))^T$. We use *vector* and *function* as synonyms. The words *coordinates* and *coefficients* are also used as synonyms. For a matrix M , the element from the i^{th} line and j^{th} column is denoted by $M(i, j)$. The support diameter of a function is called the *size* or *scale* of the function.

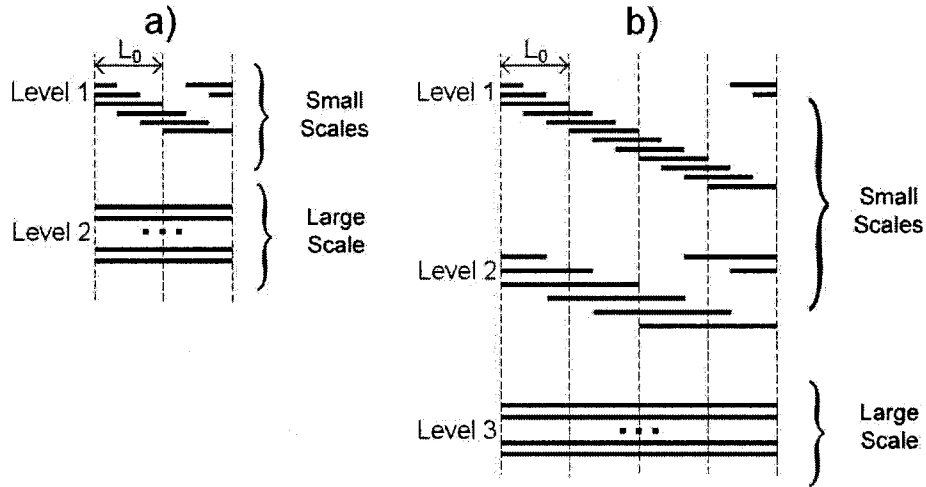


Figure 4.1: Support locations for a DOMPC basis with a) $n_{\max} = 2$ and b) $n_{\max} = 3$. The support of each vector is represented by one (or several) black segments on the same horizontal line.

4.2 Definition and properties of double overlap MPC

4.2.1 Double overlap structure of the vector supports

The locations of the supports of the basis vectors in a DOMPC basis is illustrated in figure 4.1. At the first level (figures 4.1-a and 4.1-b, top), the double overlap of the supports implies that adjacent supports are shifted by $L_0/3$ points with respect to one another. It follows that L_0 must be an integer multiple of three. At the window edges, the vectors are assumed to be *periodic*, i.e. their support overlaps both the beginning and the end of the signal. In this paper, a DOMPC with $L_0 = k$ will be called a DOMPC- k .

4.2.2 Number of degrees of freedom

The number of degrees of freedom is a measure of the basis ability to adapt to a reference signal. Let $N_{\text{DF}}(n)$ denote the number of degrees of freedom at level n , and $N_c(n)$ the number of constraints at level n . For independent constraints, the number of degrees of freedom satisfies

$$N_{\text{DF}}(n) = L_n - N_c(n).$$

For a small scale vector at level n , the constraints to be respected are normalisation, orthogonality to smaller scale vectors and orthogonality to two level- n vectors. For the k^{th} large scale vector, $1 \leq k \leq k_{\text{max}}$, the constraints are normalisation, orthogonality to all small scale vectors and orthogonality to the large scale vectors with indices $i = 1, 2, \dots, k - 1$.

At level one, each vector of size L_0 must be normalised and orthogonal to two shifted copies of itself, for a total of three constraints. At level two, we observe in figure 4.1-b that each vector of size $2L_0$ must be normalised, orthogonal to two shifted copies of itself and orthogonal to eight level-one vectors, for a total of 11 constraints. At level three, each vector of size 2^2L_0 must be normalised, orthogonal to two shifted copies of itself, and orthogonal to eight level-two vectors and 14 level-one vectors, for a total of 25 constraints. These results can be summarized as follows:

$$\begin{aligned} N_{\text{DF}}(1) &= L_0 - 3 \\ N_{\text{DF}}(2) &= 2L_0 - 3 - (6 + 2) \\ N_{\text{DF}}(3) &= 2^2L_0 - 3 - (6 + 2) - (2 \times 6 + 2). \end{aligned} \tag{4.1}$$

More generally, for $1 \leq n < n_{\max}$, $N_{\text{DF}}(n)$ takes the form

$$\begin{aligned}
 N_{\text{DF}}(n) &= 2^{n-1}L_0 - 3 - (6 + 2) - (2 \times 6 + 2) - \dots - (2^{n-2} \times 6 + 2) \\
 &= 2^{n-1}L_0 - 3 - 2(n-1) - 6(1 + 2 + \dots + 2^{n-2}) \\
 &= 2^{n-1}L_0 - 1 - 2n - 6(2^{n-1} - 1) \\
 &= 2^{n-1}(L_0 - 6) + 5 - 2n.
 \end{aligned} \tag{4.2}$$

If $L_0 \neq 6$, then the result (4.2) implies that $N_{\text{DF}}(n) \sim 2^{n-1}(L_0 - 6)$ as $n \rightarrow \infty$. For $N_{\text{DF}}(n)$ to increase as n increases, it is necessary that $L_0 > 6$. Since L_0 must also be an integer multiple of three to allow for the double overlap structure, it follows that

$$L_0 \geq 9. \tag{4.3}$$

If $L_0 \geq 9$, then it follows from (4.2) that $N_{\text{DF}}(n)$ increases as L_0 increases. One may therefore be tempted to choose a large value of L_0 to maximize the vectors adaptivity. However, we will see in section 4.2.3 that the proportion of small scale vectors also depends on L_0 and must be taken into consideration.

4.2.3 Proportion of small scale vectors

The proportion of small scale vectors (PSSV) is defined by

$$\text{PSSV}(n_{\max}) = \frac{N_{\text{SS}}(n_{\max})}{N_{\text{T}}(n_{\max})}, \tag{4.4}$$

where $N_{\text{SS}}(n_{\max})$ denotes the number of small scale vectors for a basis constructed with $n_{\max}$ levels, and $N_{\text{T}}(n_{\max})$ denotes the total number of vectors for a basis constructed with $n_{\max}$ levels.

For $n_{\max} = 2$, we observe in figure 4.1-a that there are six level-one vectors. For $n_{\max} = 3$, we observe in figure 4.1-b that there are six level-two vectors and 12 level-one vectors. For $n_{\max} = 4$, it can be shown that there are six

level-three vectors, 12 level-two vectors and 24 level-one vectors. These results can be summarized as follows:

$$\begin{aligned}
 N_{\text{SS}}(2) &= 6 \\
 N_{\text{SS}}(3) &= 6 + 2 \times 6 \\
 N_{\text{SS}}(4) &= 6 + 2 \times 6 + 2^2 \times 6.
 \end{aligned} \tag{4.5}$$

More generally, for $n_{\text{max}} \geq 2$, we may write

$$\begin{aligned}
 N_{\text{SS}}(n_{\text{max}}) &= 6 (1 + 2 + 2^2 + \dots + 2^{n_{\text{max}}-2}) \\
 &= 6 (2^{n_{\text{max}}-1} - 1).
 \end{aligned} \tag{4.6}$$

The total number of vectors in the basis is $N_{\text{T}}(n_{\text{max}}) = 2^{n_{\text{max}}-1} L_0$, hence the PSSV takes the form

$$\begin{aligned}
 \text{PSSV}(n_{\text{max}}) &= \frac{6 (2^{n_{\text{max}}-1} - 1)}{2^{n_{\text{max}}-1} L_0} \\
 &= \frac{6}{L_0} \left(1 - \frac{1}{2^{n_{\text{max}}-1}} \right).
 \end{aligned} \tag{4.7}$$

Since $n_{\text{max}} \geq 2$, it follows from (4.7) that

$$\frac{3}{L_0} \leq \text{PSSV}(n_{\text{max}}) < \frac{6}{L_0}. \tag{4.8}$$

Moreover, equation (4.7) implies that

$$\text{PSSV}(n_{\text{max}}) \sim \frac{6}{L_0} \text{ as } n_{\text{max}} \rightarrow \infty. \tag{4.9}$$

Since $L_0 \geq 9$, we conclude that DOMPC reach the same maximal PSSV than disjoint MPC [12], i.e. $6/9 = 2/3$.

4.2.4 Comparison of $N_{\text{DF}}(n)$ for disjoint MPC and DOMPC

For disjoint MPC [12] and DOMPC, the number of degrees of freedom are given by

$$\begin{aligned} N_{\text{DF}}^{(\text{disjoint})}(n) &= 2^{n-1}(L_0 - 2) + 1 \\ N_{\text{DF}}^{(\text{DOMPC})}(n) &= 2^{n-1}(L_0 - 6) + 5 - 2n. \end{aligned} \quad (4.10)$$

For disjoint MPC, it was shown [12] that L_0 must be greater or equal to three to obtain an adaptive basis (the case $L_0 = 2$ corresponds to the Haar basis). It was also shown that the maximal PSSV of $2/3$ is obtained for $L_0 = 3$. For DOMPC, we have seen that the maximal PSSV is also $2/3$ and is obtained for $L_0 = 9$.

In this paper, one of our objectives is to increase the number of degrees of freedom of the basis vectors without decreasing the PSSV. From this standpoint, it is natural to compare the functions $N_{\text{DF}}(n)$ using values of L_0 that produce comparable values of the PSSV in the two bases.

For disjoint MPC bases, it was shown that [12]

$$\lim_{n_{\text{max}} \rightarrow \infty} \text{PSSV}(n_{\text{max}}) = \frac{2}{L_0} =: \text{PSSV}^*. \quad (4.11)$$

For DOMPC bases, we have seen that

$$\lim_{n_{\text{max}} \rightarrow \infty} \text{PSSV}(n_{\text{max}}) = \frac{6}{L_0} =: \text{PSSV}^*. \quad (4.12)$$

It follows from (4.11) and (4.12) that $L_0 = 2/\text{PSSV}^*$ for disjoint MPC bases and $L_0 = 6/\text{PSSV}^*$ for DOMPC bases, where PSSV^* denotes the maximal proportion of small scale vectors. Substituting these expressions of L_0 in the

first and second lines of (4.10) respectively, we get

$$\begin{aligned} N_{\text{DF}}^{(\text{disjoint})}(n) &= 2^n \left(\frac{1}{\text{PSSV}^*} - 1 \right) + 1 \\ N_{\text{DF}}^{(\text{DOMPC})}(n) &= 3 \times 2^n \left(\frac{1}{\text{PSSV}^*} - 1 \right) + 5 - 2n. \end{aligned} \quad (4.13)$$

It is clear from (4.13) that $N_{\text{DF}}^{(\text{DOMPC})}(n)$ is approximately *three times larger* than $N_{\text{DF}}^{(\text{disjoint})}(n)$ for large values of n , which is a significant increase in the basis degree of data-adaptivity.

4.3 Construction of double-overlap MPC

4.3.1 Correlation matrix

Double overlap MPC are obtained at each level by solving an optimisation problem. The problem is to find a vector $\mathbf{X}$ that satisfies several constraints and that minimizes the quantity $\mathbb{E}\{(\mathbf{X}^T \mathbf{S})^2\}$, where $\mathbf{S}$ is the signal and $\mathbb{E}\{\dots\}$ denotes an expectation value. If the support of vector $\mathbf{X}$ is K -dimensional and the signal $\mathbf{S}$ is N -dimensional, this expectation value is estimated with the arithmetic average $\langle (\mathbf{X}^T \mathbf{S})^2 \rangle$, which is defined by

$$\langle (\mathbf{X}^T \mathbf{S})^2 \rangle = \frac{1}{N-K+1} \sum_{i=0}^{N-K} \left(\sum_{n=1}^K X(n) S(i+n) \right)^2. \quad (4.14)$$

Expanding the square sum in (4.14), it can be shown that

$$\langle (\mathbf{X}^T \mathbf{S})^2 \rangle = \mathbf{X}^T \mathcal{C}^{(K)} \mathbf{X},$$

where $\mathcal{C}^{(K)}$ is a correlation matrix defined by

$$\mathcal{C}^{(K)}(n, m) \equiv \frac{1}{N-K+1} \sum_{i=0}^{N-K} S(i+n) S(i+m). \quad (4.15)$$

4.3.2 Formulation of the optimization problem

The small scale vector $\mathbf{X}_n$, where $n < n_{\max}$, is the solution of the following optimization problem:

$$\begin{cases} \mathbf{X}_n^T C \mathbf{X}_n \text{ minimum} \\ \mathbf{X}_n^T \mathbf{X}_n = 1 \\ \mathbf{X}_n^T D_1 \mathbf{X}_n = 0 \\ \mathbf{X}_n^T D_2 \mathbf{X}_n = 0 \\ \mathbf{X}_n^T \mathbf{Z}_j = 0, \quad j = 1, \dots, m(n). \end{cases} \quad (4.16)$$

In (4.16), $C := \mathcal{C}^{(L_n)}$, where $\mathcal{C}^{(K)}$ is given by (4.15). The $\mathbf{Z}_j$ s are the vectors of level $i < n$ that have a support which overlaps the support of $\mathbf{X}_n$. The equation $\mathbf{X}_n^T \mathbf{X}_n = 1$ is the normalisation constraint. The matrices D_1 and D_2 are *symmetrical shift matrices*. The matrix D_1 is defined by

$$D_1(i, j) = \frac{1}{2} \delta_{i+\frac{1}{3}L_n, j}, \quad j \geq i. \quad (4.17)$$

The matrix D_2 is defined by

$$D_2 = D_1^2. \quad (4.18)$$

4.3.3 Solution of the optimization problem

In (4.16), the constraints

$$\mathbf{X}_n^T \mathbf{Z}_j = 0, \quad j = 1, \dots, m(n) \quad (4.19)$$

define a vector subspace $\mathcal{E}_n$ of dimension $d_n = L_n - m(n)$. Let

$$Q_n := \{\mathbf{V}_1, \mathbf{V}_2, \dots, \mathbf{V}_{d_n}\}$$

denote an orthonormal basis of $\mathcal{E}_n$. The constraints (4.19) can be taken into account by expressing $\mathbf{X}_n$ in this basis, i.e. $\mathbf{X}_n = y_1 \mathbf{V}_1 + \dots + y_{d_n} \mathbf{V}_{d_n}$, where the y_i s are the coordinates of $\mathbf{X}_n$ in the basis Q_n . The vectors $\mathbf{V}_i$, which belong to the kernel of the matrix

$$\mathbf{A}_n := (\mathbf{Z}_1 \mathbf{Z}_2 \dots, \mathbf{Z}_{d_n})^T,$$

can be obtained with an algorithm designed for kernel computation. The coordinates vector $\mathbf{Y}_n := (y_1, y_2, \dots, y_{d_n})^T$ and $\mathbf{X}_n$ are related by the change of variable

$$\mathbf{X}_n = \mathbf{P}^T \mathbf{Y}_n, \quad (4.20)$$

where $\mathbf{P}$ is a *rectangular* matrix defined by $P(i, j) \equiv V_i(j)$. The change of variables (4.20) links the coordinates of $\mathbf{X}_n$ in the two bases used, i.e. the canonical basis of $\mathbb{R}^{L_n}$ and Q_n . Note that $\mathbf{P}$ is orthogonal, i.e. $\mathbf{P}\mathbf{P}^T = \mathbf{I}$.

Using the change of variable (4.20), the problem (4.16) takes the simpler yet equivalent form

$$\left\{ \begin{array}{ll} \mathbf{Y}_n^T \overline{\mathbf{C}} \mathbf{Y}_n & \text{minimum} \\ \mathbf{Y}_n^T \mathbf{Y}_n & = 1 \\ \mathbf{Y}_n^T \overline{\mathbf{D}}_1 \mathbf{Y}_n & = 0 \\ \mathbf{Y}_n^T \overline{\mathbf{D}}_2 \mathbf{Y}_n & = 0, \end{array} \right. \quad (4.21)$$

where

$$\begin{aligned} \overline{\mathbf{C}} &:= \mathbf{P} \mathbf{C} \mathbf{P}^T \\ \overline{\mathbf{D}}_1 &:= \mathbf{P} \mathbf{D}_1 \mathbf{P}^T \\ \overline{\mathbf{D}}_2 &:= \mathbf{P} \mathbf{D}_2 \mathbf{P}^T. \end{aligned} \quad (4.22)$$

Note that $\overline{\mathbf{C}}$, $\overline{\mathbf{D}}_1$ and $\overline{\mathbf{D}}_2$ are also symmetrical.

The matrix $\overline{\mathbf{C}}$ is symmetrical therefore its eigenvectors can be chosen to form an orthonormal basis of $\mathbb{R}^{d_n}$. The orthonormal eigenvectors of $\overline{\mathbf{C}}$ will be

denoted by $\mathbf{W}_i$, $i = 1, \dots, d_n$. The corresponding eigenvalues will be denoted by λ_i , $i = 1, \dots, d_n$, with $\lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_{d_n}$.

If the last two constraints of problem (4.21) are excluded, then the problem solution is the eigenvector of $\overline{\mathbf{C}}$ having the smallest eigenvalue, i.e. $\mathbf{Y}_n = \pm \mathbf{W}_1$, for which the objective function takes the value $\mathbf{Y}_n^T \overline{\mathbf{C}} \mathbf{Y}_n = \lambda_1$. More generally, $\mathbf{W}_i^T \overline{\mathbf{C}} \mathbf{W}_i = \lambda_i$ for each i and therefore the eigenvectors $\mathbf{W}_i$ with small eigenvalues produce small values of the objective function $\mathbf{Y}_n^T \overline{\mathbf{C}} \mathbf{Y}_n$. This observation leads us to search for an approximate solution of problem (4.21) in the span of the vectors $\{\mathbf{W}_1, \mathbf{W}_2, \mathbf{W}_3\}$. In other words, we will search for a solution in the form

$$\mathbf{Y}_n = \sum_{i=1}^3 a_i \mathbf{W}_i, \quad (4.23)$$

where the a_i s are real parameters that will be adjusted to satisfy the constraints in problem (4.21) and minimize the objective function. Note that we use three vectors because problem (4.21) involves three constraints.

If the vectors $\{\mathbf{W}_1, \mathbf{W}_2, \mathbf{W}_3\}$ do not allow to satisfy the constraints in (4.21), then we try to find a solution with another triplet of eigenvectors, e.g. $\{\mathbf{W}_1, \mathbf{W}_2, \mathbf{W}_4\}$, until a satisfactory solution is found. To minimize the objective function, these attempts are performed on triplets $\{\mathbf{W}_i, \mathbf{W}_j, \mathbf{W}_k\}$ which are sorted according to increasing values of the sum $\lambda_i + \lambda_j + \lambda_k$, triplets with small values of this sum being selected first. Scanning triplets $\{\mathbf{W}_i, \mathbf{W}_j, \mathbf{W}_k\}$ in that order, the first triplet encountered which satisfies the constraints (4.21) is selected as the solution.

For a given triplet of eigenvectors, the approximate solution of the optimization problem (4.21) is obtained as follows. Substituting (4.23) into (4.21) leads

to

$$\left\{ \begin{array}{l} \sum_{i=1}^3 \lambda_i a_i^2 \text{ minimum} \\ \sum_{i=1}^3 a_i^2 = 1 \\ \sum_{i=1}^3 \sum_{j=1}^3 d_{i,j}^{(1)} a_i a_j = 0 \\ \sum_{i=1}^3 \sum_{j=1}^3 d_{i,j}^{(2)} a_i a_j = 0 \end{array} \right. \quad (4.24)$$

where $d_{i,j}^{(k)} := \mathbf{W}_i^T \overline{\mathbf{D}}_k \mathbf{W}_j$, $k = 1, 2$. Note that the matrices $d^{(k)}$, $k = 1, 2$, are symmetrical.

The last three equations of (4.24) can be written in the explicit form

$$\left\{ \begin{array}{l} a_1^2 + a_2^2 + a_3^2 = 1 \\ d_{1,1}^{(1)} a_1^2 + 2d_{1,2}^{(1)} a_1 a_2 + 2d_{1,3}^{(1)} a_1 a_3 + d_{2,2}^{(1)} a_2^2 + 2d_{2,3}^{(1)} a_2 a_3 + d_{3,3}^{(1)} a_3^2 = 0 \\ d_{1,1}^{(2)} a_1^2 + 2d_{1,2}^{(2)} a_1 a_2 + 2d_{1,3}^{(2)} a_1 a_3 + d_{2,2}^{(2)} a_2^2 + 2d_{2,3}^{(2)} a_2 a_3 + d_{3,3}^{(2)} a_3^2 = 0. \end{array} \right. \quad (4.25)$$

Substituting a_3^2 for $1 - a_1^2 - a_2^2$ in the last two equations of (4.25), and then solving the resulting two equations for a_3 leads to

$$\left\{ \begin{array}{l} a_3 = \frac{-d_{2,2}^{(1)} a_2^2 - d_{3,3}^{(1)} (1 - a_1^2 - a_2^2) - d_{1,1}^{(1)} a_1^2 - 2d_{1,2}^{(1)} a_1 a_2}{2d_{1,3}^{(1)} a_1 + 2d_{2,3}^{(1)} a_2} =: f(a_1, a_2) \\ a_3 = \frac{-d_{2,2}^{(2)} a_2^2 - d_{3,3}^{(2)} (1 - a_1^2 - a_2^2) - d_{1,1}^{(2)} a_1^2 - 2d_{1,2}^{(2)} a_1 a_2}{2d_{1,3}^{(2)} a_1 + 2d_{2,3}^{(2)} a_2} =: h(a_1, a_2). \end{array} \right. \quad (4.26)$$

The equality $f(a_1, a_2) = h(a_1, a_2)$ leads to the equation

$$\begin{aligned}
 P_3(a_1, a_2) &:= \left(-d_{2,2}^{(1)} a_2^2 - d_{3,3}^{(1)} (1 - a_1^2 - a_2^2) - d_{1,1}^{(1)} a_1^2 - 2 d_{1,2}^{(1)} a_1 a_2 \right) \\
 &\quad \times \left(2d_{1,3}^{(2)} a_1 + 2d_{2,3}^{(2)} a_2 \right) \\
 &\quad - \left(-d_{2,2}^{(2)} a_2^2 - d_{3,3}^{(2)} (1 - a_1^2 - a_2^2) - d_{1,1}^{(2)} a_1^2 - 2 d_{1,2}^{(2)} a_1 a_2 \right) \\
 &\quad \times \left(2 d_{1,3}^{(1)} a_1 + 2d_{2,3}^{(1)} a_2 \right) \\
 &= 0,
 \end{aligned} \tag{4.27}$$

where P_3 is degree three polynomial in the variables a_1 and a_2 . For a fixed value of a_1 , the three roots of $P_3(a_1, a_2) = 0$ are denoted by

$$a_2 = g_i(a_1), \tag{4.28}$$

$i = 1, 2, 3$. For each value of a_1 , the *real* roots (4.28) are sorted according to $g_1(a_1) \leq g_2(a_1) \leq g_3(a_1)$. In general, real values of a_2 do not exist for all $a_1 \in [-1, 1]$.

Substituting (4.28) into the first equation of (4.26) yields

$$a_3 = f(a_1, g_i(a_1)), \tag{4.29}$$

$i = 1, 2, 3$. Since (4.29) and $a_1^2 + a_2^2 + a_3^2 = 1$ must be satisfied simultaneously, then the equality $|f(a_1, g_i(a_1))| = \sqrt{1 - a_1^2 - (g_i(a_1))^2}$ must be satisfied by a_1 .

We conclude that the acceptable values of a_1 must satisfy

$$F_i(a_1) = 0, \tag{4.30}$$

where the functions F_i are defined by

$$F_i(a_1) = |f(a_1, g_i(a_1))| - \sqrt{1 - a_1^2 - (g_i(a_1))^2}, \tag{4.31}$$

$i = 1, 2, 3$.

We solve equation (4.30) separately for each i . It follows from $a_1^2 + a_2^2 + a_3^2 = 1$ that $a_1 \in [-1, 1]$. To solve (4.30) for a given value of i , we sample the function $F_i(a_1)$ for regularly spaced values of a_1 (using an increment of 0.05) in the range $[-1, 1]$ and we locate the values of a_1 for which $F_i(a_1)$ changes sign, if they exist. For each interval in which $F_i(a_1)$ changes sign, we compute a precise estimate of the root using bisection. For each root of $F_i(a_1) = 0$, we compute a_2 and a_3 with (4.28) and (4.29). If several possible values of a_1 are obtained, then we choose the one which minimizes the objective function $\sum_{i=1}^3 \lambda_i a_i^2$.

It may happen that real values of a_2 cannot be obtained for any $a_1 \in [-1, 1]$ or that roots of $F_i(a_1)$ cannot be found in $[-1, 1]$, which means that it is impossible to satisfy the constraints in the span of the three eigenvectors selected. In this case, the estimation of the coefficients a_i must be repeated using a different triplet of eigenvectors.

4.3.4 Examples of DOMPC

In figure 4.2, we display the small scale basis vectors obtained with $L_0 = 15$ and $n_{\max} = 5$, using the sinusoid displayed in figure 4.3 as a reference signal. This basis is composed of four levels of small scale vectors and 150 large scale vectors.

4.4 Description of the reference signals

In the following sections, we will perform data compression and noise filtering experiments on the reference signals displayed in figure 4.3. Note that only a fraction of each complete reference signal is displayed in figure 4.3. The reference signals used contain 20000 points. Each complete reference signal is used to estimate the correlation matrices from which DOMPC are computed.

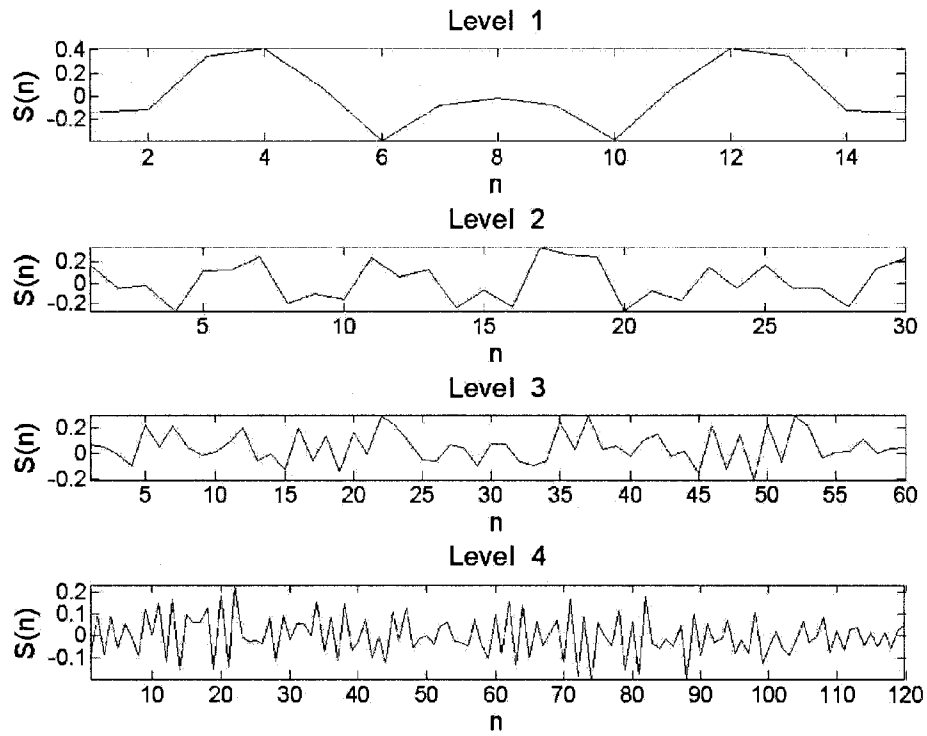


Figure 4.2: Small scale vectors for a DOMPC basis obtained with $L_0 = 15$ using a sinusoidal signal of period 30 points as a reference signal.

The sinusoidal signal (figure 4.3-a) has a 30 points period. This signal is analytical and can therefore be approximated locally by a polynomial. Wavelets should therefore allow for an efficient compression of this signal.

The square wave (figure 4.3-b) has a six points period. This is an example of a discontinuous periodic signal.

The triangular wave (figure 4.3-c) has an eight points period. This is an example of a continuous periodic signal having a discontinuous derivative.

The long-period oscillating signal (figure 4.3-d) is defined by

$$S(n) = \sum_{i=1}^3 (a_i \sin(\omega_i n) + b_i \cos(\omega_i n)) ,$$

where $a_i = 1/i$, $b_i = 1/i^2$, $\omega_i = \frac{2\pi}{T_i}$, $T_1 = 5$, $T_2 = 7$ and $T_3 = 13$. This oscillating signal is periodic with period $T = T_1 T_2 T_3 = 445$.

The random signal in figure 4.3-e was obtained by smoothing a white noise built with mutually independent random variables uniformly distributed on $[-0.5, 0.5]$. Smoothing was performed with a 200 points moving average.

The signal in figure 4.3-f was obtained from a two-dimensional fingerprint image (figure 4.4) by setting side-by-side adjacent horizontal lines of the image, using the alternate scanning directions left-to-right and right-to-left, thus preserving the signal continuity.

The signal in figure 4.3-g is a digital recording of instrumental music (percussion, piano and violin).

The signal in figure 4.3-h is a synthetic piecewise constant signal. The plateaus heights are random variables uniformly distributed on $[-0.5, 0.5]$. The plateaus horizontal widths are exponential random variables with mean 30 points. Heights and widths are independent random variables.

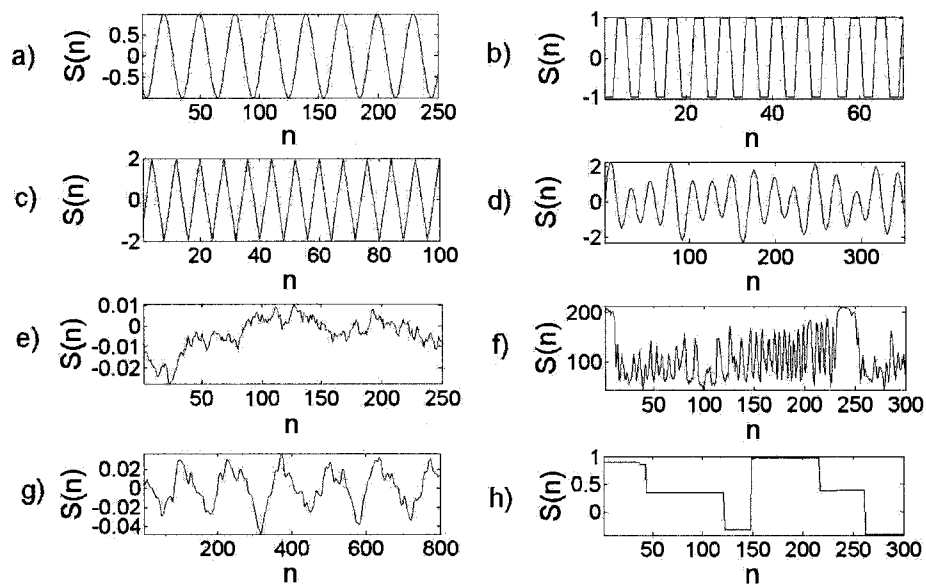


Figure 4.3: Samples of the reference signals used in this paper: a) sinusoidal signal with a 30 points period; b) square wave with a six points period; c) triangular wave with an eight points period; d) long-period oscillating signal; e) smoothed white noise; f) one-dimensional section in a fingerprint image; g) instrumental music signal; h) piecewise constant signal.

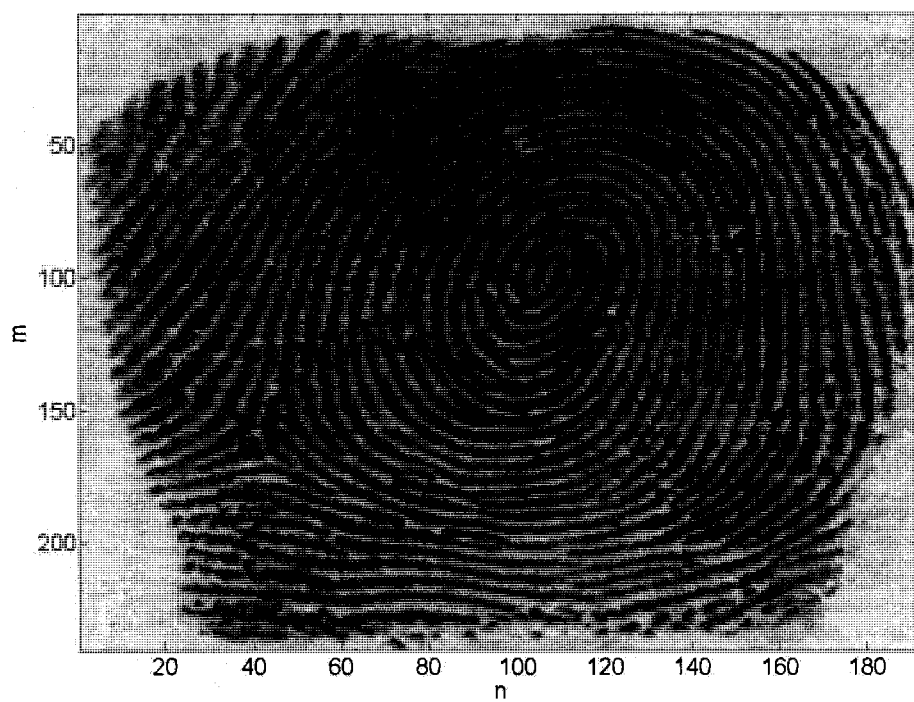


Figure 4.4: A fingerprint image.

4.5 Data compression experiments

4.5.1 Data compression method

For a signal expressed in an orthonormal basis, data compression can be performed by thresholding the signal coordinates (also called coefficients). Thresholding is a non-linear filtering operation which consists in setting to zero all the coefficients which are smaller in absolute value than a given threshold.

If the signal $\mathbf{S}$ is expressed in a DOMPC basis $\{\mathbf{X}_i, i = 1, \dots, N\}$, then the coefficient c_i associated to the vector $\mathbf{X}_i$ is computed with $c_i = \mathbf{X}_i^T \mathbf{S}$. If the signal $\mathbf{S}$ is expressed in a wavelet basis, then the coefficients are computed with the wavelet transform. In this paper, the wavelet transforms were computed with the *wavelet* toolbox available in the scientific computing software *Matlab*, using the functions *wavedec* and *waverec*, which perform the transform and the reconstruction respectively. We used the *periodic* wavelet transform because our DOMPC bases are also periodic. The wavelet transforms are performed via iterative high-pass and low-pass filterings of the signal, as described by Mallat [11].

After thresholding the coefficients, the signal is reconstructed from the modified coefficients, which yields a reconstructed signal $\mathbf{S}_R \neq \mathbf{S}$. To quantify the reconstruction quality, we use the signal-to-noise ratio (SNR) which is defined in units of decibels by

$$\text{SNR} = 10 \log_{10} \left(\frac{\|\mathbf{S}\|^2}{\|\mathbf{S} - \mathbf{S}_R\|^2} \right).$$

For a given basis (DOMPC or wavelet), we compute the SNR for the signal reconstructed from the N_C largest coefficients (in absolute value). We characterize the basis compression performance with the plot displaying the SNR versus the compression ratio $100 N_C/N$.

4.5.2 Data compression results

In this section, we compare the data compression efficiency obtained with DOMPCs and wavelets. We used the bases DOMPC-9 and DOMPC-15, for which $L_0 = 9$ and $L_0 = 15$ respectively. We chose the wavelets Daubechies-4 and Daubechies-8 because their support diameters (8 and 16 respectively) are comparable to the support diameters of the two DOMPC bases used.

The data compression results for the eight reference signals are presented in figure 4.5 (sinusoid, square and triangular waves, long-period oscillating signal) and figure 4.6 (smoothed white noise, fingerprint, music recording, piecewise constant signal).

In figure 4.5, we observe that the DOMPC bases perform much better than wavelets. For instance, with the sinusoidal signal (figure 4.5-a), the bases DOMPC-15 and DOMPC-9 achieve a nearly perfect reconstruction quality ($\text{SNR} > 250$ dB) with 1% and 7% of the coefficients respectively. In contrast, a 100 dB reconstruction with wavelets requires about 50% of the coefficients. Similar results are obtained with the square wave (figure 4.5-b), the triangular wave (figure 4.5-c) and the long-period oscillating signal (figure 4.5-d). For the square and triangular waves, we observe in the figures 4.5-b and 4.5-c that the wavelet bases performance is poor, which is explained by the occurrence of frequent discontinuities in the signal and its derivative.

In figure 4.6, we observe that the compression performance of DOMPC and wavelets are more comparable. For the smoothed white noise (figure 4.6-a) and the fingerprint (figure 4.6-b), the performances are comparable for $100 N_C/N > 4$ %. For the music signal (figure 4.6-c), the basis DOMPC-15 performs significantly better than the best wavelet. For $100 N_C/N \in [10\%, 80\%]$, we observe a 10-15 dB difference in the SNR, which corresponds roughly to a factor ten. For the piecewise continuous signal (figure 4.6-d), DOMPC bases and wavelets

produce similar performances for $100 N_C/N < 12\%$. However, for larger values of $100 N_C/N$, the wavelet Daubechies-4 outperforms all the other bases by achieving a perfect reconstruction ($\text{SNR} > 225 \text{ dB}$) with 40% of the coefficients.

4.5.3 Interpretation

To understand why DOMPC bases perform much better than wavelets for the signals analyzed in figure 4.5, it is useful to examine the coefficients scale in the curves SNR versus $100 N_C/N$. More precisely, it will be interesting to see if the coefficients are associated to large or small scale vectors. In figure 4.7, we plotted these curves separately for the four bases, using the sinusoid as a reference signal. Coefficients attached to large scale and small scale vectors are marked by crosses (+) and dots (.) respectively. Note that for wavelets, small and large scale coefficients correspond to detail and approximation coefficients respectively.

In figure 4.7-b, which corresponds to the best performing basis DOMPC-15, we observe that only two large scale coefficients capture almost all the signal energy and allow for a nearly perfect reconstruction (250 dB). This is a case where small scale vectors in the DOMPC-15 basis manage to be exactly orthogonal to the reference signal. Using wavelets, a comparable level of orthogonality can only be achieved using much higher order wavelets with correspondingly larger support sizes.

In the figures 4.7-c and 4.7-d, which correspond to the wavelet bases, we observe that the SNR increases more slowly as $100 N_C/N$ increases and that mostly small scale coefficients are used. This indicates that the signal energy is spread more evenly among basis vectors. It follows that a much larger number of coefficients is required to achieve a quality reconstruction.

For the other signals analyzed in figure 4.5 (square and triangular wave,

long-period oscillating signal), the superior performance of DOMPC bases with respect to wavelets is explained by the same phenomenon: the DOMPC small scale vectors manage to be perfectly orthogonal to the reference signal and all the signal energy is captured by a few large scale coefficients.

Consider now the case of the piecewise constant signal, for which wavelets perform significantly better than DOMPCs. For the wavelets (figures 4.8-c and 4.8-d), we observe that the proportion of vanishing coefficients is approximately 50% (actually 60% and 40% for Daubechies-4 and Daubechies-8 respectively). It follows that an excellent signal reconstruction can be achieved with approximately 50% of the coefficients. For DOMPCs (figures 4.8-a and 4.8-b), we observe that large coefficients are located mostly in the large scales. In fact, nearly all the signal energy is spread among large scale vectors, which is unfavorable to data-compression via thresholding. We observe that most vanishing coefficients are associated to small scale vectors and we found that approximately 50% of small scale coefficients are negligible. According to equation (4.9), we know that at most $2/3 \approx 67\%$ and $2/5 \approx 40\%$ of the DOMPC coefficients are small scale coefficients for DOMPC-9 and DOMPC-15 respectively. If we neglect all vanishing small scale coefficients, a good reconstruction can be achieved for DOMPC-9 and DOMPC-15 with approximately $100 - 0.5 \times 67 = 67\%$ and $100 - 0.5 \times 40 = 80\%$ of the coefficients respectively. These percentages correspond to the compression ratios observed in the figures 4.8-a and 4.8-b for an SNR of 75 dB, which marks the frontier between small and large scale coefficients.

We conclude that the poorer compression performance of DOMPC for the piecewise signal is caused by two main factors: the relatively even distribution of the signal energy among large scale vectors and the lower PSSV. Note that a lower PSSV is not a handicap if most large scale vectors are orthogonal to the

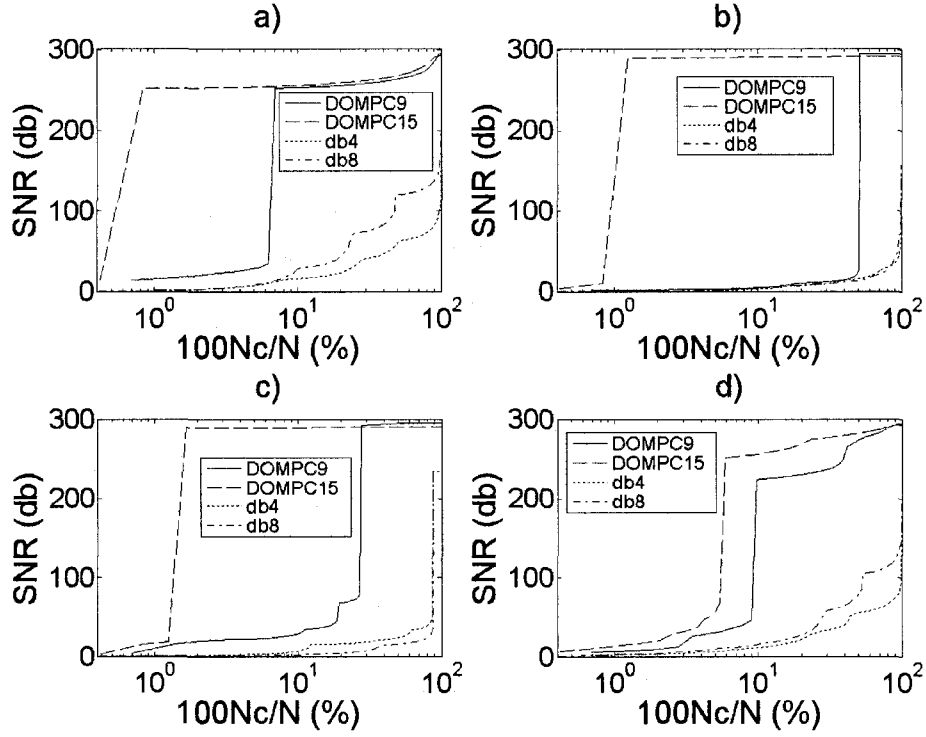


Figure 4.5: SNR versus $100 N_C/N$ for a) the sinusoid; b) the square wave; c) the triangular wave; d) the long-period oscillating signal.

signal, as we observed for the four signals analysed in figure 4.5.

4.6 Noise filtering experiments

4.6.1 Noise filtering method

We consider a signal S which is corrupted by an additive noise B , resulting in a noisy signal $S' := S + B$. We assume that S' is represented in an orthonormal basis. If some of the basis vectors, say $\{V_i, i = 1, 2, \dots, i_{\max}\}$, are orthogonal to S , then

$$V_i^T S' = V_i^T (S + B) = V_i^T B.$$

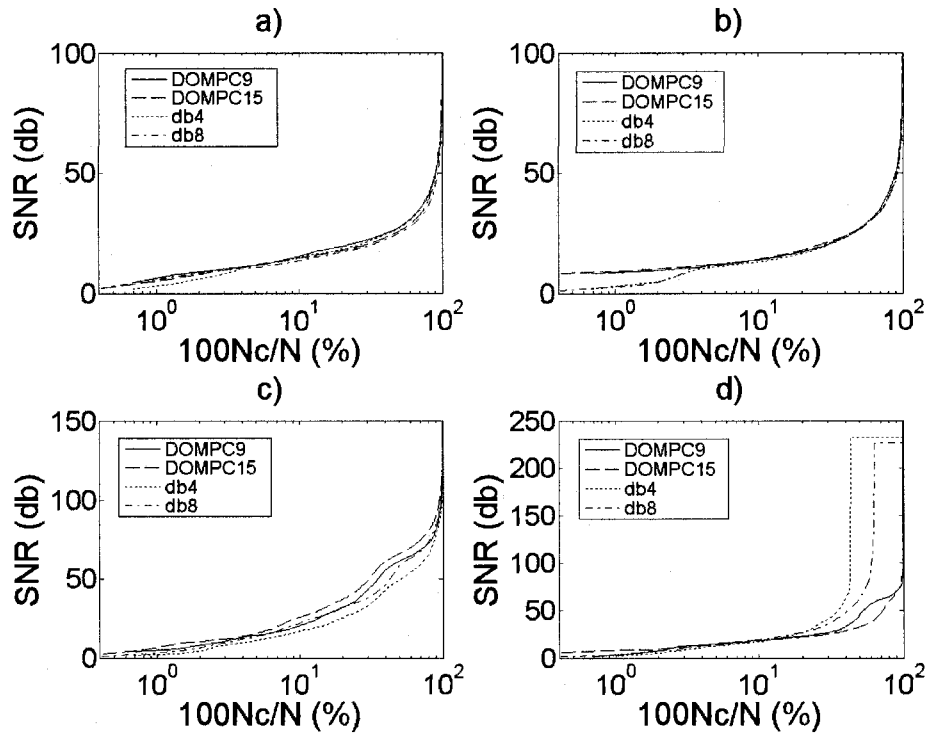


Figure 4.6: SNR versus $100N_c/N$ for a) the smoothed white noise; b) the fingerprint signal; c) the music recording; d) the piecewise constant signal.

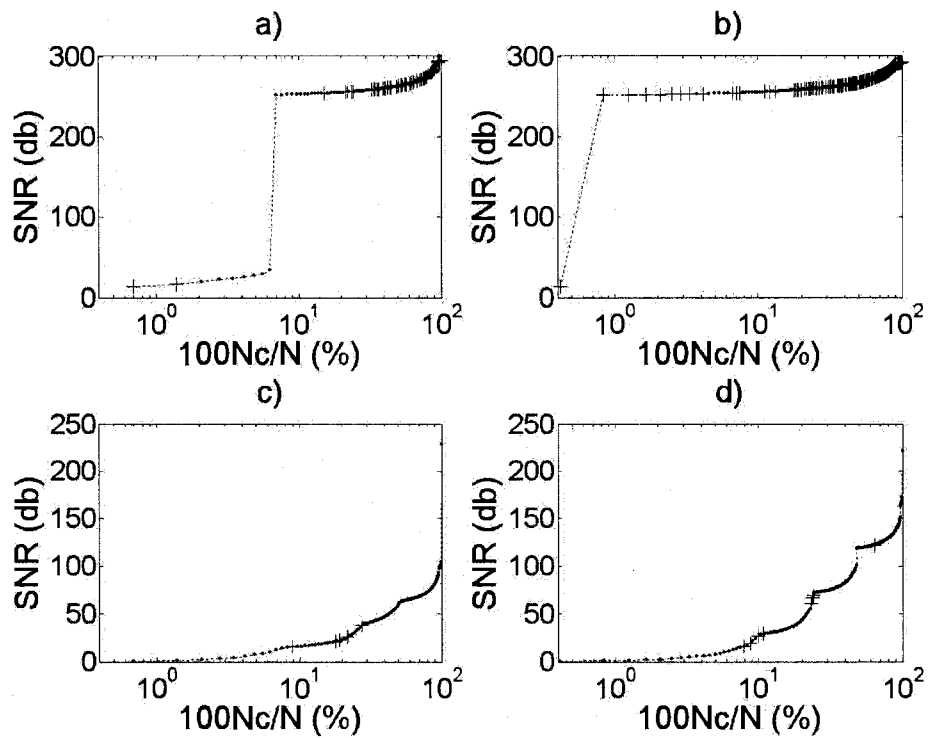


Figure 4.7: SNR versus $100 N_C/N$ for the sinusoidal signal. Coefficients attached to large scale and small scale vectors are marked by crosses (+) and dots (.) respectively. The bases used are a) DOMPC-9; b) DOMPC-15; c) Daubechies-4; d) Daubechies-8.

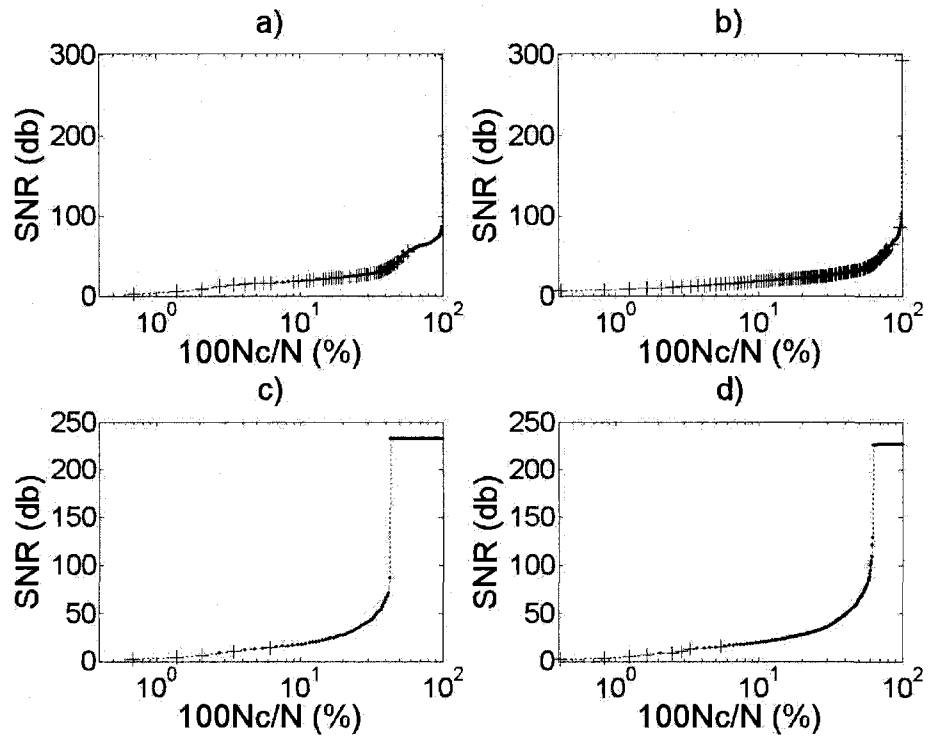


Figure 4.8: SNR versus $100 N_c/N$ for the piecewise constant signal. Coefficients attached to large scale and small scale vectors are marked by crosses (+) and dots (.) respectively. The bases used are a) DOMPC-9; b) DOMPC-15; c) Daubechies-4; d) Daubechies-8.

Hence the coefficients $c_i := \mathbf{V}_i^T \mathbf{S}'$ contain only the energy of the additive noise. It follows that setting these c_i s to zero increases the signal to noise ratio. This noise removal approach can be used for the coefficients obtained either with the DWT or the DOMPC bases.

For all the filtering experiments presented here, we use a noise of the form

$$B(n) = \frac{\alpha}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\left(\frac{n-\mu}{\sigma}\right)^2} W(n), \quad (4.32)$$

$n = 0, 1, \dots, N-1$. The noise defined by (4.32) is a uniformly distributed zero-mean and unit variance white noise $W(n)$ which is multiplied by gaussian localisation function of mean $\mu = (N-1)/2$ and standard deviation $\sigma = N/8$. The noise is therefore localized primarily around the window center. The factor $\alpha = 50$ controls the noise maximal amplitude. As an example, we display in figure 4.9 the sinusoidal signal $\mathbf{S}$, the noise $\mathbf{B}$ and the noisy signal $\mathbf{S}' = \mathbf{S} + \mathbf{B}$.

We will test two types of filtering. The first one, called *hard thresholding*, consists in setting to zero all the coefficients which are smaller (in absolute value) than a given threshold. The second one, called *shrinking thresholding*, consists in transforming the coefficients c_i into modified coefficients c'_i according to the rule

$$c'_i := \begin{cases} 0 & \text{if } |c_i| \leq \tau \\ \text{sign}(c_i) (|c_i| - \tau) & \text{if } |c_i| > \tau, \end{cases} \quad (4.33)$$

where τ denotes a threshold.

For both methods, an optimal threshold was selected as following. Filtering was performed for 100 regularly spaced thresholds in the range $[0, 1.5 \sigma \sqrt{2 \log(N)}]$. In this range, the threshold providing the largest SNR was selected. In this context, the SNR is defined in units of decibels by

$$\text{SNR} = 10 \log_{10} \left(\frac{\|\mathbf{S}\|^2}{\|\mathbf{S} - \mathbf{S}_F\|^2} \right).$$

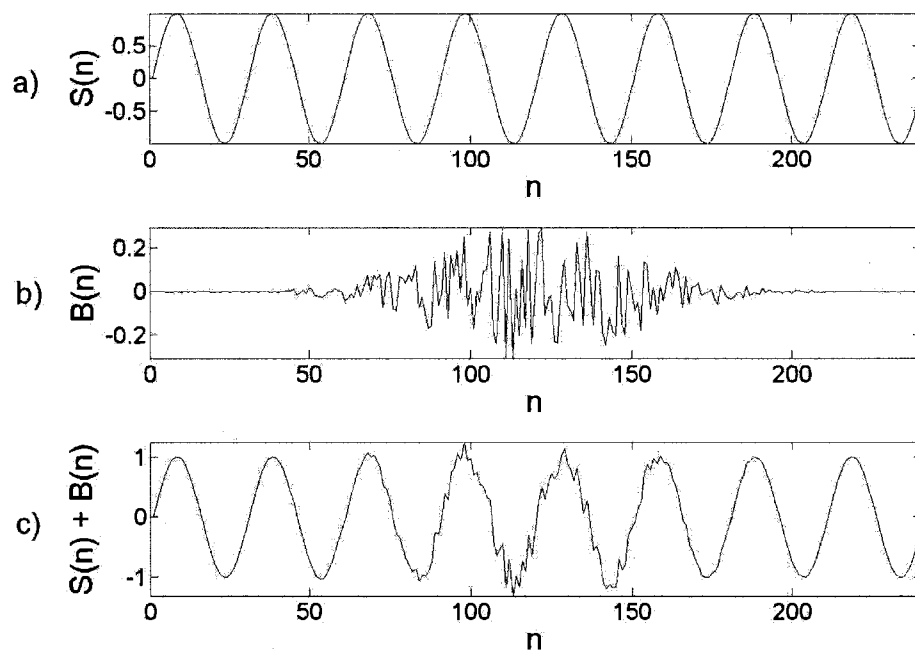


Figure 4.9: Example of the sinusoidal signal (a), the localized synthetic noise (b) and their sum (c).

where S is the original noise-free signal and S_F denotes the filtered noisy signal S' . Note that $\sigma\sqrt{2\log(N)}$ is the optimal threshold proposed by Donoho [3] for the filtering of additive white Gaussian noise on piecewise polynomial signals. Our noise is not gaussian and is localised according to (4.32), hence this optimal threshold does not correspond exactly to our experiments. Nevertheless, we found that it provides a useful reference point for our grid search of the optimal threshold.

4.6.2 Noise filtering results

The filtering results for the sinusoid and the square wave are displayed in the figures 4.10 and 4.11 respectively. These figures display the residual signal E , which is defined by $E = S - S_F$, together with the corresponding SNR. The wavelets used (Daubechies-8, coiflet-3, symlet-8) were selected because their width is comparable to the smallest support size for the basis DOMPC-15, i.e. 15 points.

In figure 4.10, we observe that hard thresholding always produces a larger SNR than shrinking thresholding. The best results are obtained with the basis DOMPC-15, which produces an SNR of 46.6 dB. This SNR is more than twice larger than the SNR obtained with the best wavelet basis (22.8 dB for the basis symlet-8 with hard thresholding). We also observe that hard thresholding produces a residual signal which is distributed rather homogeneously in the window, whereas shrinking thresholding produces a residual signal which is more intense in the window center.

For the square wave signal, we observe in figure 4.11 that hard thresholding again produces better results than shrinking thresholding, although the SNR improvement is almost negligible for wavelets. The basis DOMPC-15 gives an excellent SNR of 45 dB, which is again twice larger than the best wavelet SNR

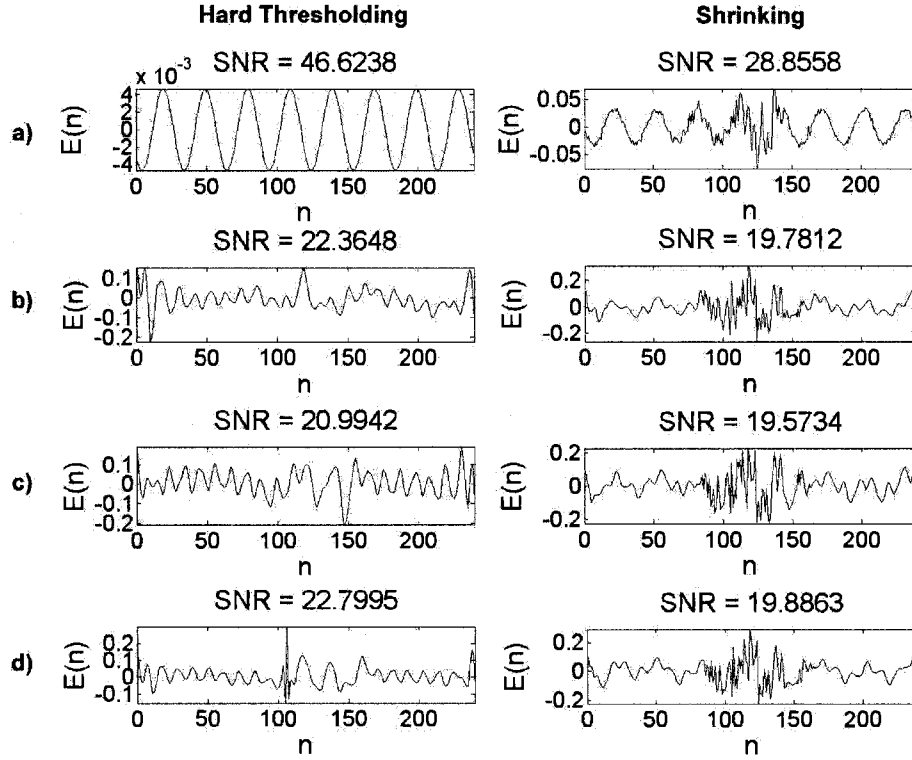


Figure 4.10: Filtering residue $E = S - S_F$ for the noisy sinusoidal signal using hard thresholding (left column) and shrinking thresholding (right column). The corresponding SNR is given on top of each plot in units of decibels. The bases used are a) DOMPC-15, b) Daubechies-8, c) coiflet-3 and d) symlet-8.

(21.95 dB for the basis symlet-8 with hard thresholding). For this signal, we observe that both filtering methods produce a residual signal intensity which is concentrated in the window center.

For the piecewise continuous signal, we found that there is no significant difference in the performances of the basis DOMPC-15 or the wavelets: they produced SNR values in the range [10 dB, 12 dB], which is a rather poor quality filtered signal.

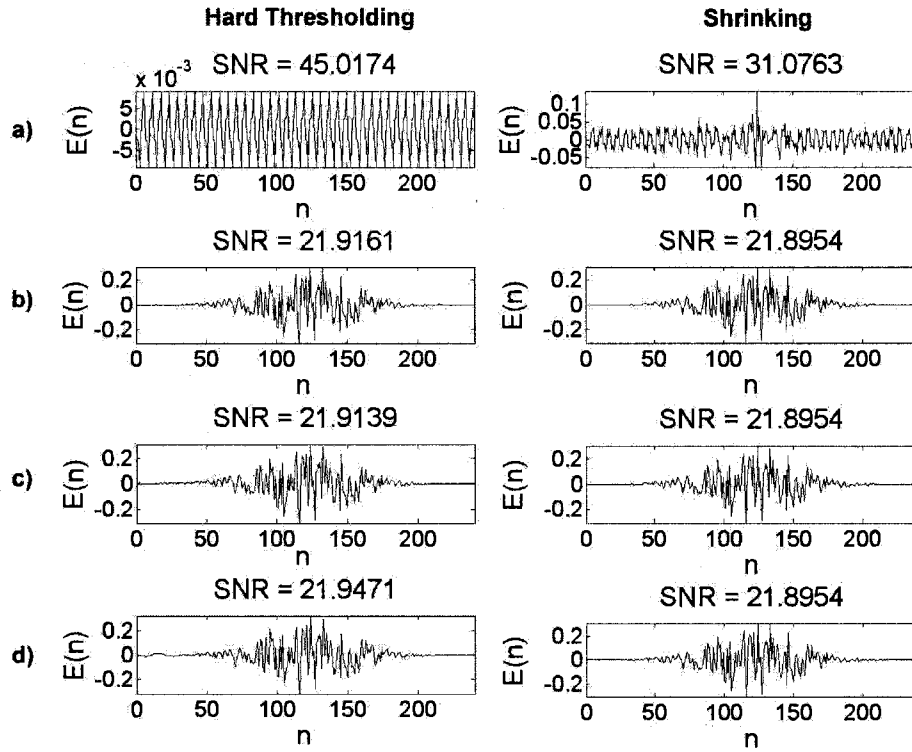


Figure 4.11: Filtering residue $E = S - S_F$ for the noisy square wave signal using hard thresholding (left column) and shrinking thresholding (right column). The corresponding SNR is given on top of each plot in units of decibels. The bases used are a) DOMPC-15, b) Daubechies-8, c) coiflet-3 and d) symlet-8.

4.7 Conclusion

Comparing disjoint MPC bases and DOMPC bases having the same proportion of small scale vectors, we have shown that the number of degrees of freedom for a basis vector is about three time larger for a DOMPC basis than for a disjoint MPC basis. Hence DOMPC are significantly more data-adaptive than disjoint MPC.

In our data compression experiments, we found that DOMPC can perform much better than wavelets for *low dimensional* signals, i.e. signals that can be expressed as a linear combination of a few constant vectors (e.g. square or triangular wave signals, sinusoids, large-period oscillating signals). For such signals, small scale DOMPC vectors can be optimized to be exactly orthogonal to the signal if the minimum support size L_0 is large enough. Exact orthogonality of small scale vectors to the signal is obtained because DOMPC vectors can achieve orthogonality to a limited number of vectors. In such cases, DOMPC can perform much better than wavelets having a comparable support size. Moreover, DOMPC can produce excellent results even for non-smooth signals, contrary to wavelets. It is worth noting that the compression performance for a digitized instrumental music signal was found to be significantly better with DOMPC than with wavelets. In the future, it will be worth investigating this results to determine if a similar compression gain can also be obtained with different types of music.

For piecewise continuous signals, we found that wavelets performed significantly better than DOMPC in data compression experiments. In this case, the lower performance of DOMPC bases is caused primarily by they lower proportion of small scale vectors.

For the sinusoid and the square wave signal, our noise filtering experiments (using hard thresholding and shrinking thresholding) showed that DOMPC per-

formed much better wavelets. For these two signals, the basis DOMPC-15 produced an SNR more than twice larger than wavelets having a comparable support size. For the piecewise constant signal, both DOMPC and wavelets produced comparable and rather poor filtering performances, with an SNR in the range 10-12 dB. Finally, we found that hard thresholding appears to perform better than shrinking thresholding.

From signal processing applications which require an efficient representation of localised patterns, e.g. spikes or singularities, it can be advantageous to have a large proportion of small scale vectors. For such applications, DOMPC are less competitive than wavelets because both disjoint MPC and DOMPC have a proportion of small scale vectors which is at most $2/3$. However, it may be possible to develop new construction approaches for MPC bases to increase the proportion of small scale vectors, while maintaining data adaptability.

Acknowledgements The authors would like to thank the professors Charles Audet and François Soumis, from École Polytechnique de Montréal, for their valuable suggestions related to the optimization problem examined in section 4.3.3. A. Saucier acknowledges financial support from NSERC.

Bibliography

- [1] M. Aminghafari, N. Cheze, and J.-M. Poggi. Multivariate denoising using wavelets and principal component analysis. *Computational Statistics and Data Analysis*, 50:2381–2398, 2006.
- [2] B.R. Bakshi. Multiscale pca with application to multivariate statistical process monitoring. *AIChE J.*, 44:1596–1610, 1998.
- [3] D. L. Donoho and I. M. Johnstone. Ideal spatial adaptation via wavelet shrinkage. *Biometrika*, 81:425–455, 1994.
- [4] Z. Geng and A. Zhu. A wavelet-based adaptive mspca for process signal monitoring and diagnosis. In *International Conference on Information Acquisition*, pages 135–139, June 21-25 2004. ISBN: 0-7803-8629-9.
- [5] J. S. Geronimo, D. P. Hardin, and P. R. Massopust. Fractal functions and wavelet expansions based on several scaling functions. *Journal of Approximation Theory*, 78(3):373–401, 1994.
- [6] E. C. Karvounis and D.I. Fotiadis. Maternal and fetal heart rate extraction from abdominal recordings using multi-scale principal components analysis. In *29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 6507–6510, Aug 22-26 2007.

- [7] R. E. Learned and A. S. Willsky. A wavelet packet approach to transient signal classification. *Appl. Comput. Harmonic Anal.*, 2(3):265–278, 1995.
- [8] J. M. Lilly and J. Park. Multiwavelet spectral and polarization analyses of seismic records. *Geophys. J. Int.*, 122(3):1001–1021, 1995.
- [9] S. Mallat and Z. Zhang. Singularity detection and processing with wavelets. *IEEE Transactions on Signal Processing*, 41(12):3397–3415, 1993.
- [10] Stéphane Mallat. *A wavelet tour of signal processing*. Academic Press, 525 B Street, Suite 1900, San Diego, CA 92101-4495, USA, 1998. (a) section 4.3. (b) section 7.2.1. (c) section 10.1.1. (d) section 9.1.3.
- [11] Stéphane Mallat. *A wavelet tour of signal processing*. Academic Press, 525 B Street, Suite 1900, San Diego, CA 92101-4495, USA, 1998.
- [12] A. Saucier. Construction of data-adaptive orthogonal wavelet bases with an extension of principal component analysis. *Applied and computational harmonic analysis*, 18:300–328, 2005.
- [13] Pascal Yiou, Didier Sornette, and Michael Ghil. Data-adaptive wavelets and multi-scale singular spectrum analysis. *Physica D*, 142:254–290, 2000.

CHAPITRE 5 MÉTHODE DU « ZIPPER »

Dans ce chapitre, on décrit la première méthode de résolution qui fut explorée dans le but de résoudre le problème d'optimisation

$$\begin{cases} X'CX \text{ min} \\ X'X = 1 \\ X'D_1X = 0 \\ X'D_2X = 0 \\ X'Z_j = 0, j = 1, \dots, m \end{cases} \quad (5.1)$$

tel que décrit dans l'article au chapitre 4.

Un algorithme itératif a été élaboré pour résoudre ce problème d'optimisation. À la $i^{\text{ème}}$ itération, on trouve le vecteur X_i qui résout le problème d'optimisation

$$\begin{cases} X'_iCX_i \text{ min} \\ X'_iX_i = 1 \\ X'_iD_1X_{i-1} = 0 \\ X'_iD_2X_{i-2} = 0 \\ X'_iZ_j = 0, j = 1, \dots, m \end{cases} \quad (5.2)$$

À la première itération ($i = 0$), on génère deux vecteurs aléatoires pour les vecteurs X_{-1} et X_{-2} . On itère jusqu'à ce que le vecteur X_i converge. La position des supports des vecteurs sur un même niveau est représentée dans la figure (5.1 a)

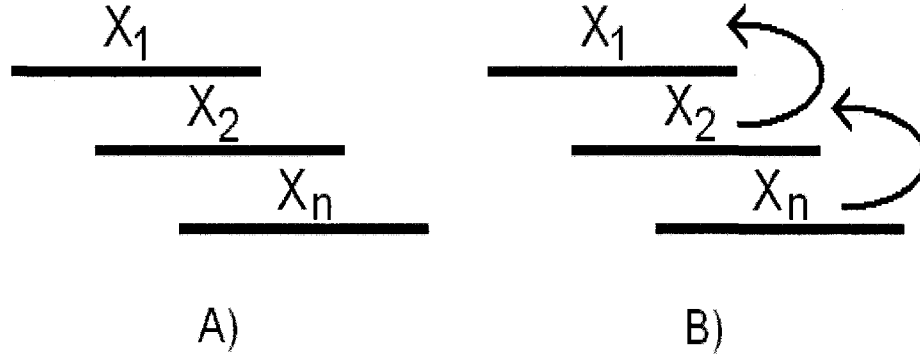


Figure 5. 1 Position des supports des vecteurs des petites échelles sur un même niveau (A) et substitutions à effectuer à chaque itération pour la méthode itérative (B)

On cherche un vecteur X_i qui est orthogonal aux vecteurs X_1 et X_2 . On peut donc ajouter ces deux vecteurs à l'ensemble des vecteurs Z_j auxquels le vecteur X_i doit être orthogonal. On résout ensuite le problème d'optimisation en le projetant dans le sous-espace vectoriel orthogonal aux vecteurs Z_j en effectuant le changement de variable

$$X_i = P' Y_i \quad (5.3)$$

où P est la matrice définie par $[P]_{i,j} = V_i(j)$ et $Q = \{V_1, \dots, V_{d_n}\}$ est une base orthonormale du sous-espace vectoriel orthogonal aux vecteurs Z_j . Avec ce changement de variable, le problème d'optimisation (5.2) devient

$$\begin{cases} Y_i' \bar{C} Y_i \min \\ Y_i' Y_i = 1 \end{cases} \quad (5.4)$$

où $\bar{C} := PCP'$. Ce nouveau système d'équations est un problème de composantes principales dont la solution est le vecteur propre de la matrice $\bar{C}$

associé à la plus petite valeur propre. On trouve ainsi la solution optimale Y_i que l'on projette dans l'espace original pour trouver la solution recherchée X_i .

C'est ici que la méthode itérative entre en jeu. On effectue les substitutions suivantes

$$\begin{aligned} X_1 &\leftarrow X_2 \\ X_2 &\leftarrow X_n \end{aligned} \tag{5.5}$$

telles qu'illustrées dans la figure (5.1 b). On itère par la suite jusqu'à ce que l'algorithme converge. Si la solution X_i converge vers les vecteurs solutions des itérations précédentes X_{i-1} et X_{i-2} , les contraintes quadratiques du problème d'optimisation (5.2) deviennent $X_i' D_1 X_i = 0$ et $X_i' D_2 X_i = 0$ ce qui fournit une solution au problème initial (5.1).

La pratique a montré que cet algorithme fournit de bons résultats lorsqu'il converge. Cependant, il ne converge pas toujours ce qui fournit dans ces cas une solution qui ne respecte pas les contraintes quadratiques d'orthogonalité. Après plusieurs tentatives pour régler ce problème, cette méthode a été mise de côté au profit d'une autre qui est décrite dans l'article du chapitre 4.

CHAPITRE 6 SYNTHÈSE et CONCLUSION

6.1 Discussion

La première méthode explorée, le *Zipper*, semblait être une bonne piste pour résoudre le problème d'optimisation à cause de la simplicité de cet algorithme. Lorsque la convergence est obtenue, la méthode produit des résultats de bonne qualité. Néanmoins, les expériences ont montré que la convergence n'est pas garantie. Cette absence de convergence produit des résultats qui ne respectent pas les contraintes quadratiques d'orthogonalité. À cause de ce manque de fiabilité, cette méthode a été écartée au profit d'une nouvelle méthode complètement différente.

La deuxième méthode consiste à approximer la solution dans un sous-espace vectoriel engendré par une combinaison de trois vecteurs propres de la matrice de corrélation du signal calculée à différentes échelles. Les bases DOMPC formées à partir de cette méthode offrent des performances en compression et en filtrage qui dépassent de beaucoup celles des bases d'ondelettes pour les signaux de faible dimension (sinusoïde, onde carrée, onde triangulaire et signal oscillant de longue période). Ces bonnes performances se résument par le fait que quelques vecteurs de grande échelle des bases DOMPC recueillent l'ensemble de l'énergie du signal. Pour les autres signaux testés, les bases DOMPC offrent des performances en compression semblables à celles des bases d'ondelettes. On note le cas prometteur de la compression d'un extrait de musique pour lequel les bases DOMPC performant significativement mieux que les ondelettes. Pour le signal constant par morceaux, les bases d'ondelettes fournissent des performances en compression et en filtrage supérieures à celles des bases DOMPC. Ces derniers résultats s'expliquent par la faible proportion de vecteurs de petites échelles et parce que l'énergie est répartie dans plusieurs vecteurs de

grande échelle. Les bases DOMPC ont l'avantage d'être adaptatives au signal contrairement aux bases d'ondelettes pour lesquelles il est difficile de prévoir quelle ondelette est optimale pour compresser ou filtrer un signal. La méthode utilisée pour générer des bases DOMPC donne néanmoins une solution approximative du problème d'optimisation étant donné que la recherche de cette solution s'effectue dans un sous-espace vectoriel tridimensionnel.

6.2 Conclusion et recommandations

Dans ce projet, deux approches ont été testées pour augmenter l'adaptabilité des bases MPC tout en conservant la proportion de vecteurs de petites échelles. La méthode itérative manque de fiabilité et a été mise de côté. L'autre méthode qui consiste à approximer la solution a permis d'atteindre l'objectif d'augmenter l'adaptabilité des bases générées comparativement aux bases MPC tout en préservant la proportion de vecteurs de petites échelles.

Il serait avantageux d'explorer plus en détails dans quelle mesure les bases DOMPC sont efficaces pour compresser des extraits de musique et si les bons résultats de compression peuvent être obtenus pour d'autres types de musique.

Une autre structure des supports des vecteurs offre des perspectives d'amélioration intéressantes. En effet, l'ajout de plusieurs couches de vecteurs sur un même niveau permettrait de contrôler l'adaptabilité des bases MPC ainsi que la proportion de vecteurs de petites échelles. Cette méthode ne nécessite plus de chevauchement entre les supports ce qui rend le problème d'optimisation beaucoup plus simple à résoudre. Cela permettrait de trouver une solution optimale au problème d'optimisation plutôt qu'une approximative.

RÉFÉRENCES BIBLIOGRAPHIQUES

- [1] AMINGHAFARI, M., CHEZE, N., and POGGI, J.-M. (2006). Multivariate denoising using wavelets and principal component analysis. *Computational Statistics and Data Analysis*, 50: 2381–2398.
- [2] BAKSHI, B. R. (1998). Multiscale PCA with application to multivariate statistical process monitoring. *AIChE J.*, 44: 1596-1610.
- [3] DAUBECHIES, I. (1992). *Ten lectures on wavelets*. SIAM, Philadelphia (PA).
- [4] DONOHO, D. L. and JOHNSTONE, I. M. (1994). Ideal spatial adaptation via wavelet shrinkage. *Biometrika*, 81: 425–455.
- [5] FAHMY, M.F., HASAN, Y. M. Y. and MOHAMED, M. F. (2005). Orthogonal signal decomposition with applications in wavelet denoising. In *Proceedings of the Twenty-Second National Radio Science Conference*, 2005. NRSC 2005., pages 293-300. March 15-17.
- [6] GENG, Z. and ZHU, A. (2004). A wavelet-based adaptative MSPCA for process signal monitoring and diagnosis. In *International Conference on Information Acquisition*, pages 135-139, June 21-25. ISBN: 0-7803-8629-9.
- [7] GERONIMO, J. S., HARDIN, D. P., and MASSOPUST, P. R. (1994). Fractal functions and wavelet expansions based on several scaling functions. *Journal of Approximation Theory*, 78(3): 373–401.

- [8] JOLLIFFE, I. T. (1986). *Principal component analysis*, Springer-Verlag, New York.
- [9] KARVOUNIS, E. C. and FOTIADIS, D.I. (2007). Maternal and fetal heart rate extraction from abdominal recordings using multi-scale principal components analysis. In *29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 6507–6510, Aug 22-26.
- [10] LEARNED, R. E. and WILLSKY, A. S. (1995). A wavelet packet approach to transient signal classification. *Appl. Comput. Harmonic Anal.*, 2(3): 265–278.
- [11] LILLY, J. M. and PARK, J. (1995). Multiwavelet spectral and polarization analyses of seismic records. *Geophys. J. Int.*, 122(3): 1001–1021.
- [12] MALLAT, S. and ZHANG, Z. (1993). Singularity detection and processing with wavelets. *IEEE Transactions on Signal Processing*, 41(12): 3397–3415.
- [13] MALLAT, S. (1998). *A wavelet tour of signal processing*. Academic Press, 525 B Street, Suite 1900, San Diego, CA 92101-4495, USA.
- [14] PARK, D. and LEE, M. H. (1994). Image compression based on best wavelet packet bases. *IEEE Transactions on Consumer Electronics*, 40(3): 527-537.

- [15] PEARSON, K. (1901). On lines & planes of closest fit to systems of points in space, *philosophical Magazine*, S. 6. Vol. 2: 559-572 No. 11 Nov.
- [16] RAJPOOT, N. M., WILSON, R. G., MEYER, F. G. and COIFMAN, R. R. (2003). Adaptive wavelet packet basis selection for zerotree image coding. *IEEE Transactions on Image Processing*, 12(12): 1460-1472.
- [17] SARDY, S., TSENG, P. and BRUXE, A. (2001). *Robust wavelet denoising*, Signal Processing, IEEE transactions on, June 2001, vol. 49, Issue 6, pages 1146-1152.
- [18] SAUCIER, A. (2005). Construction of data-adaptive orthogonal wavelet bases with an extension of principal component analysis. *Applied and computational harmonic analysis*, 18: 300-328.
- [19] SHEN-EN, Q. and HOLLINGER, A. B. (2000). Applications of wavelet data compression using modified zerotrees in remotely sensed data, *Geoscience and Remote Sensing Symposium*, 2000, 6: 2654-2656.
- [20] STRANG, G. and NGUYEN, T. (1996). *Wavelets and Filter Banks*, Wellesley-Cambridge Press, Box 812060, Wellesley MA 02181 USA. Revised in 1997.
- [21] USEVITCH, B. E. (2001). A tutorial on modern lossy wavelet image compression: foundations of JPEG 2000, *Signal Processing Magazine IEEE*, Sept. 2001, vol. 18, Issue 5, pages 22-35.

- [22] YIOU, P., SORNETTE, D., and GHIL, M. (2000). Data-adaptive wavelets and multi-scale singular spectrum analysis. *Physica D*, 142: 254–290.

ANNEXE

ANNEXE A : CODE *MATLAB* DE LA MÉTHODE DE RÉOLUTION PAR APPROXIMATION

Dans cette annexe, on présente l'ensemble des fichiers *Matlab* formés lors de la réalisation de ce projet.

Voici les fichiers nécessaires à la formation des DOMPC :

```
%=====
% FICHIER : DOMPC.m
% DESCRIPTION : Ce fichier permet de générer une base DOMPC-N adaptée
%               au signal contenu dans le fichier FichierSignal. Cette base DOMPC-N
%               se forme de levelMax-1 niveaux de petites échelles et d'un niveau de
%               grande échelle.
% PARAMÈTRES :
%   FichierSignal : nom du fichier .mat contenant le signal (variable : signal).
%   N : Taille du support du vecteur de la plus petite échelle.
%   levelMax : niveau maximal de décomposition (grande échelle).
% SORTIES :
%   BaseComplete : vecteurs de la base DOMPC-N (vecteurs colonnes).
%   L : vecteur donnant le nombre de vecteurs par niveau (du niveau 1 jusqu'au
%       nombre de vecteurs de grande échelle.
%   MatricesC : matrices de corrélation du signal calculées à différentes échelles
%=====
```

```
function [BaseComplete,L,MatricesC] = DOMPC(FichierSignal,N,levelMax)
```

```
% On charge le signal (variable: signal)
```

```
load(FichierSignal);
```

```
% Pas à la plus petite échelle
```

```

pas = N/3;
% --- Génération des vecteurs de petite échelle --- %

for level = 1:(levelMax-1)
    % affichage à l'écran : niveau courant
    disp('-----+')
    disp(['niveau' num2str(level)])
    % Initialisation
    ContOrthoNivInf = [];
    ContOrthoTot = [];
    C = [];
    % Facteur d'échelle
    FactEch = 2^(level-1);
    % Calcul de la matrice de corrélation au niveau actuel
    C = MatriceCexacte(signal,N,FactEch);
    MatricesC{level} = C;
    % On forme les vecteurs des niveaux inférieurs
    if(level>1)
        for i=1:level-1
            ContOrthoNivInf = DecalVecteur(vectDOMPC{i},FactEch*N,2^(i-1)*pas);
            % On normalise les vecteurs
            for s = 1:size(ContOrthoNivInf,2)
                ContOrthoNivInf(:,s) = ContOrthoNivInf(:,s)/norm(ContOrthoNivInf(:,s));
            end
            ContOrthoTot = [ContOrthoTot ContOrthoNivInf];
        end
    end
    % Nullspace des vecteurs des niveaux inférieurs
    P = null(ContOrthoTot');
    % Orthonormalisation de Pmod
    P = orth(P);
    % Initialisation de P pour niveau 1 (matrice Identité)
    if(level == 1)
        P = eye(FactEch*N);
    end
end

```

```

end

% Recherche de la meilleure solution approximative
Xbest = SolveOptiProblem(N,C,P,level);

% affichage à l'écran : donnée sur solution trouvée
disp(['Nb points: ' num2str(length(Xbest))])
disp(['Nb contraintes ortho. niv. inf. : ' num2str((size(ContOrthoTot,2)))]
disp(['Dimension espace ortho aux contraintes: ' num2str(size(P,2))])
Ndg(level) = size(P,2)-3;
disp(['NB DEGRES DE LIBERTE: ' num2str(Ndg(level))])
disp(['Xt*C*X = ' num2str(Xbest'*C*Xbest) ])
disp(['Norme: ' num2str(norm(Xbest))])

% Affectation de solution trouvée
vectDOMPC{level} = Xbest;

% Fonction qui test l'orthogonalité intra-niveau
MaxTestOrthoIntra = TestOrthoIntra(vectDOMPC{level},pas,level,N);
disp(['Produit scalaire max. intra-niveau: ' num2str(MaxTestOrthoIntra)])
% Fonction qui test l'orthogonalité inter-niveau
if(level>1)
    MaxTestOrthoInter = TestOrthoInter(vectDOMPC,pas,level,N);
    disp(['Produit scalaire max. inter-niveau: ' num2str(MaxTestOrthoInter)])
end

% Graphique des vecteurs de la DOMPC
figure(1)
subplot(levelMax-1,1,level)
plot(vectDOMPC{level},'k')
title(['Level ' num2str(level)],'FontSize',20)
axis([1 length(vectDOMPC{level}) min(vectDOMPC{level}) ...
      max(vectDOMPC{level})])
xlabel('n','fontsize',20)
ylabel('S(n)','fontsize',20)

```

```

fh = figure(1);
set(fh, 'color', 'white');
end

% --- Assemblage de la base vectorielle --- %

% affichage à l'écran
disp('-----')
disp('Formation de la base complete')
disp(['Nombre de points: ' num2str(2^(levelMax-1)*N)])
% Initialisation
BaseComplete = [];
L = [];
% Génération de la sortie BaseComplete
for lvl = 1:(levelMax-1)
    VectBorder = [];
    BaseNiv = DecalVecteur(vectDOMPC{lvl}, 2^(levelMax-1)*N, 2^(lvl-1)*pas);
    % On rassemble les vecteurs couvrant les bordures
    for b = 1:(N/pas)-1
        VectBorder(:,b) = BaseNiv(:,b)+BaseNiv(:,end-(N/pas)+b+1);
    end
    BaseNiv = [BaseNiv(:,N/pas:end-N/pas+1), VectBorder];
    disp(['Vecteurs niveau' num2str(lvl) ': ' num2str(size(BaseNiv,2))])
    BaseComplete = [BaseComplete, BaseNiv];
    % Vecteur contenant le nb de vecteurs à chaque niveau
    L = [L; size(BaseNiv,2)];
end
disp(['Nb Vecteurs total: ' num2str(size(BaseComplete,2))])

% --- Génération des vecteurs de grande échelle --- %

NbGHMax = size(BaseComplete,1)-size(BaseComplete,2);
% Ajout du nombre de vecteurs de grande échelle dans L
L = [L; NbGHMax];

```

```

% affichage à l'écran : vecteur L et nombre vect. grande échelle
disp(['Vecteur L: ' num2str(L')])
disp('-----')
disp(['NB Vect. Grande echelle: ' num2str(NbGHMax)])
% Calcul de la proportion de vecteurs de petites échelles
NsSurN = (N*2^(levelMax-1)-NbGHMax)/(N*2^(levelMax-1));
% affichage à l'écran : proportion de vecteurs de petites échelles
disp(['Ns/N: ' num2str(NsSurN)])

% Facteur d'échelle (grande échelle)
FactEch = 2^(levelMax-1);

% Fonction d'autocorrélation du signal (grande échelle)
C = MatriceCexacte(signal,N,FactEch);
MatricesC{levelMax} = C;

% On forme les vecteurs de grande échelle un à un
for index = 1:NbGHMax
    % Nullspace des vecteurs de petites échelles et grande échelle
    Pmod = null(BaseComplete');
    % Orthonormalisation de Pmod
    Pmod = orth(Pmod);
    % On projecte la matrice C
    Cmod = Pmod'*C*Pmod;
    % On trouve les vecteurs et valeurs propres de Cmod
    [VectP,D] = eig(Cmod);
    % On choisit les valeurs propres sur la diagonale
    ValP = D(find(D~=0));
    % Conserve le vect. propre associé la valeur propre la plus petite.
    y2 = VectP(:,(find(ValP==min(ValP))));
    % On revient dans l'espace vect. départ
    Vnew = real(Pmod*y2);
    Vnew = Vnew/norm(Vnew);
    % On inverse le vecteur si sa plus grande valeur en absolu est négative

```

```
if(abs(min(Vnew)) > abs(max(Vnew)))  
    Vnew = -Vnew;  
end  
% On ajoute le vecteur de grande échelle dans la base  
BaseComplete = [BaseComplete, Vnew];  
end  
  
% Vérification que BaseComplete est orthonormale  
VerifBaseComplete(BaseComplete);  
%===== %  
% FIN du fichier DOMPC.m  
%===== %
```

```

%=====
% FICHIER : MatriceCexacte.m
% DESCRIPTION : Ce fichier calcule la matrice de corrélation du signal
%               pour la formation d'une base DOMPC-N avec un facteur
%               d'échelle donné.
% PARAMÈTRES :
%       signal : variable contenant le signal
%       N : Taille du support du vecteur de la plus petite échelle.
%       FactEch : Facteur d'échelle en fonction du niveau
% SORTIES :
%       C : Matrice de corrélation du signal
%=====

function C = MatriceCexacte(signal,N,FactEch)

% Taille du signal
LS = length(signal);
% Calcule de la matrice de corrélation
for x = 1:FactEch*N
    for y = 1:FactEch*N
        S1 = signal(1+x-1:LS-FactEch*N+x);
        S2 = signal(1+y-1:LS-FactEch*N+y);
        C(x,y) = S1'*S2;
    end
end
C = C/(LS-FactEch*N+1);
%=====
% FIN du fichier MatriceCexacte.m
%=====

```

```
%=====
% FICHIER : DecalVecteur.m
% DESCRIPTION : Ce fichier permet de retourner un ensemble de vecteurs
%               correspondant à différents décalage d'une Vecteur sur
%               une plage de point avec un pas donné.
% PARAMÈTRES :
%   Vecteur : Vecteur décallé.
%   plage : Plage de point sur laquelle le vecteur est décallé.
%   pas : Pas de décalage.
% SORTIES :
%   out : Ensemble des vecteurs correspondant aux décalage d'une
%   Vecteur
%=====
```

```
function out = DecalVecteur(Vecteur,plage,pas)
```

```
% Taille du vecteur
N = length(Vecteur);
% Initialisation
i=1;

% On forme les décalage du vecteur
while(i*pas <= plage+N-1)
    % Aucun zero a gauche
    if(i*pas <= N)
        out(:,i) = [Vecteur(N-i*pas+1:N) ; zeros(plage-i*pas,1)];
    % zeros a gauche et a droite
    elseif(i*pas <= plage)
        out(:,i) = [zeros(i*pas-N,1) ; Vecteur ; zeros(plage-i*pas,1)];
    % Aucun zero a droite
    else
        out(:,i) = [zeros(i*pas-N,1) ; Vecteur(1:plage-i*pas+N)];
    end
    % increment de l'index
```

```
    i = i+1;  
end  
%=====0%  
% FIN du fichier DecalVecteur.m  
%=====0%
```

```
%=====
% FICHER : SolveOptiProblem.m
% DESCRIPTION : Ce fichier retourne la meilleure approximation au problème
%               d'optimisation
% PARAMÈTRES :
%   N : Taille du support du vecteur de la plus petite échelle.
%   C : Matrice de corrélation du signal
%   P : Base orthonormale du sous-espace orthogonal aux vecteurs niv. inf.
% level : Niveau
% SORTIES :
%   Xbest : Meilleure approximation de la solution
%=====
```

```
function [Xbest] = SolveOptiProblem(N,C,P,level)
```

```
% Index de recherche pour les combinaisons de triplet de vect. propres
```

```
indexSearch = 0;
```

```
% Pas du decallage
```

```
pas = N/3;
```

```
% Pas d'échantillonnage
```

```
ech = 0.05;
```

```
% Facteur d'échelle
```

```
factEch = 2^(level-1);
```

```
% On initie XtCX à l'infinie
```

```
XtCXBest = inf;
```

```
% On forme les matrices de decallage D1 et D2
```

```
D1temp = diag(ones(factEch*pas,1),factEch*(N-pas))+...
```

```
           diag(ones(factEch*pas,1),-factEch*(N-pas));
```

```
D2temp = diag(ones(factEch*2*pas,1),factEch*(N-...
```

```
           2*pas))+diag(ones(factEch*2*pas,1),-factEch*(N-2*pas));
```

```
% On projette les matrices dans le sous-espace ortho.
```

```
D1 = P'*D1temp*P;
```

```

D2 = P'*D2temp*P;
Cmod = P'*C*P;
% Calcul des vecteurs et valeurs propres
[VectP,D] = eig(Cmod);
ValP = D(find(D~=0));
% On met les vect. p. en ordre croissant par rapport aux valeurs propres
[ValP,I] = sort(ValP,'ascend');
for q=1:length(I)
    VectPsort(:,q) = VectP(:,I(q));
end

% index des 3 vecteurs propres utilisees
indexVPTemp = ListeVectPropres(size(VectPsort,2));
% On trie la liste en fonction de la somme des valeurs propres
indexVPTemp(4,:) =
real(ValP(indexVPTemp(1,:))+ValP(indexVPTemp(2,:))+ValP(indexVPTemp(3,:)));
indexVPTot = sortrows(indexVPTemp',4);
indexVPTot = indexVPTot(:,1:3)';

% Recherche de la solution approximative
disp('..... Recherche d"une solution .....')
while(XtCXBest == inf)

    % On cherche parmi les 5 triplets de vecteurs propres
    indexVP = indexVPTot(:,1+5*indexSearch:5+5*indexSearch);
    % Increment de l'index dans la liste des vect. p.
    indexSearch = indexSearch+1;

    for Q = 1:size(indexVP,2)
        % Initialisations
        X11 = [];
        X12 = [];
        X13 = [];
        FX11 = [];

```

```

FX12 = [];
FX13 = [];
% Vecteurs propres utilisees pour la combinaison
V(:,1) = VectPsort(:,indexVP(1,Q));
V(:,2) = VectPsort(:,indexVP(2,Q));
V(:,3) = VectPsort(:,indexVP(3,Q));
% Plage de A(1) = -1:1
A1 = -1:ech:1;
% Coefficients de l'equation:
%  $a_{11}A(1)^2 + a_{12}A(1)A(2) + a_{22}A(2)^2 + a_{13}A(1)A(3) + a_{23}A(2)A(3) + a_{33}A(3)^2 = 0$ 
a11 = V(:,1)'*D1*V(:,1);
a12 = 2*V(:,1)'*D1*V(:,2);
a13 = 2*V(:,1)'*D1*V(:,3);
a22 = V(:,2)'*D1*V(:,2);
a23 = 2*V(:,2)'*D1*V(:,3);
a33 = V(:,3)'*D1*V(:,3);
% Coefficients de l'equation:
%  $b_{11}A(1)^2 + b_{12}A(1)A(2) + b_{22}A(2)^2 + b_{13}A(1)A(3) + b_{23}A(2)A(3) + b_{33}A(3)^2 = 0$ 
b11 = V(:,1)'*D2*V(:,1);
b12 = 2*V(:,1)'*D2*V(:,2);
b13 = 2*V(:,1)'*D2*V(:,3);
b22 = V(:,2)'*D2*V(:,2);
b23 = 2*V(:,2)'*D2*V(:,3);
b33 = V(:,3)'*D2*V(:,3);
% Coefficients  $d*x^2 + c*x + a = 0$ ;
d = (a33*b23 + b22*a23 - a22*b23 - b33*a23);
c = (-a12*A1*b23 + a33*b13*A1 + b12*A1*a23 - a22*b13*A1 - ...
      b33*a13*A1 + b22*a13*A1);
b = (-a11*A1.^2*b23 - a12*A1.^2*b13 + b33*a23 + a33*A1.^2*b23...
      + b12*A1.^2*a13 - a33*b23 + b11*A1.^2*a23 - b33*A1.^2*a23);
a = -a11*A1.^3*b13 + b33*a13*A1 - a33*b13*A1 - ...
      b33*A1.^3*a13 + a33*A1.^3*b13 + b11*A1.^3*a13;

```

```

% Pour chacun des intervalles de A1
for h = 1:length(A1)
    % On forme le polynôme
    poly = [d c(h) b(h) a(h)];
    % On trouve les racines de ce polynôme
    racines = roots(poly);
    % On classe ces racines en ordre croissant
    racines = sort(racines);
    % Trois solutions pour A(2)
    for j = 1:3
        % A(2) = f(A(1))
        if(length(racines) == 3)
            A2 = racines(j);
        else
            % On néglige la solution
            A2 = 2;
        end
        % A(3) = f(A(1),A(2))
        if((a13*A1(h)+a23*A2) ~= 0)
            A3 = (-a33-a11*A1(h)^2+a33*A1(h)^2-a12*A1(h)*A2-...
                a22*A2^2+a33*A2^2)/(a13*A1(h)+a23*A2);
            A32 = sqrt(1-A1(h)^2-A2^2) * sign(A3);
        else
            % On néglige la solution
            A3 = 2;
        end
        % Si A(2) et A(3) sont reels et que leurs val. abs. < 1
        % On forme les fonction Fi
        if(isreal(A2) && isreal(A3) && abs(A2) < 1 && abs(A3) < 1)
            if(j==1)
                X11 = [X11 A1(h)];
                FX11 = [FX11 (A3-sign(A3)*sqrt(1-A1(h)^2-A2^2))];
            elseif(j==2)
                X12 = [X12 A1(h)];
            end
        end
    end
end

```

```

        FX12 = [FX12 (A3-sign(A3)*sqrt(1-A1(h)^2-A2^2))];
    else
        X13 = [X13 A1(h)];
        FX13 = [FX13 (A3-sign(A3)*sqrt(1-A1(h)^2-A2^2))];
    end
end
end
end

% Pour chacune des fonctions Fi
for iteFct = 1:3
    % Affectation de la bonne fonction Fi
    if(iteFct ==1)
        x1 = X11;
        Fx1 = FX11;
    elseif(iteFct ==2)
        x1 = X12;
        Fx1 = FX12;
    else
        x1 = X13;
        Fx1 = FX13;
    end
    % Bornes potentielles des zéros dans la fonction Fi
    bornesZeros = PassageZero(x1,Fx1,ech);
    % Pour chacune des bornes potentielles
    for ite0 = 1:size(bornesZeros,1)
        % On effectue une bisection
        A1 = bissec(a11,a12,a13,a22,a23,a33,b11,b12,b13,b22,b23,b33,...
            bornesZeros(ite0,1),bornesZeros(ite0,2),iteFct);
        % On reforme le polynome
        d = (a33*b23+b22*a23-a22*b23-b33*a23);
        c = (-a12*A1*b23+a33*b13*A1+b12*A1*a23-a22*b13*A1-...
            b33*a13*A1+b22*a13*A1);
        b = (-a11*A1^2*b23-a12*A1^2*b13+b33*a23+a33*A1^2*b23+...

```

```

        b12*A1^2*a13-a33*b23+b11*A1^2*a23-b33*A1^2*a23);
a = -a11*A1^3*b13+b33*a13*A1-a33*b13*A1- ...
        b33*A1^3*a13+a33*A1^3*b13+b11*A1^3*a13;
poly = [d c b a];
% on trouve les racines correspondantes
racines = roots(poly);
racines = sort(racines);
% On affecte la valeur de la racine à a2 en fonction de la
% fonction Fi en cours de recherche
A2 = racines(iteFct);
A3 = sqrt(1-A1^2-A2^2)*sign((-a33-a11*A1^2+a33*A1^2-a12*A1*A2- ...
        a22*A2^2+a33*A2^2)/(a13*A1+a23*A2));
% Si A(2) et A(3) sont reels et que leurs val. abs. < 1
% On forme les fonctions Fi
if(isreal(A2) && isreal(A3) && abs(A2) < 1 && abs(A3) < 1)
    % On forme la solution dans le SEL ortho. aux vect.
    % niv. inf.
    XDansP = A1*V(:,1)+A2*V(:,2)+A3*V(:,3);
    % Retour dans l'espace vect. original
    X = P*XDansP;
    % On inverse le vecteur si sa plus grande valeur en absolu est negative
    if(abs(min(X)) > abs(max(X)))
        X = -X;
    end
    X = X/norm(X);
    ProdScalIntra = TestOrthoIntra(X,pas,level,N);
    % Si la solution est meilleure que précédentes
    % et que la solution respecte contraintes quadratiques
    if(abs(X'*C*X) < abs(XtCXBest) && ProdScalIntra < 10e-15)
        Xbest = real(X);
        XtCXBest = X'*C*X;
    end
end
end
end
end

```

```
        end
    end
end
%=====
% FIN du fichier SolveOptiProblem.m
%=====
```

```

%=====
% FICHER : ListeVectPropres.m
% DESCRIPTION : Ce fichier retourne les combinaisons d'index possible
%               pour la sélection des triplets de vect. propres.
% PARAMÈTRES :
%       indexMax : index maximum
% SORTIES :
%       indexVP : triplet des index distincts de vect. p.
%=====

function indexVP = ListeVectPropres(indexMax)

index1 = 1;
index2 = 2;
index3 = 3;
indexVP = [];

while(index1 <= indexMax-2)
    indexVP = [indexVP [index1 index2 index3]'];
    index3 = index3+1;
    if(index3 > indexMax)
        index2 = index2+1;
        index3 = index2+1;
        if(index2 > indexMax-1)
            index1 = index1+1;
            index2 = index1+1;
            index3 = index2+1;
        end
    end
end

%=====
% FIN du fichier ListeVectPropres.m
%=====

```

```

%=====
% FICHIER : PassageZero.m
% DESCRIPTION : Ce fichier permet de donner des bornes potentielles
%               pour les zéros des fonctions Fi
% PARAMÈTRES :
%       x : coordonnées en X (var. ind.)
%       y : coordonnées en Y (var. dép.)
%       ech : pas d'échantillonnage
% SORTIES :
%       bornesZeros : couples de bornes potentielles de zéros
%=====
function bornesZeros = PassageZero(x,y,ech)

bornesZeros = [];
for j = 1:(length(y)-2)
    if(sign(y(j)) ~= sign(y(j+1)) && abs(x(j)-x(j+1)) <= 1.5*ech)
        bornesZeros = [bornesZeros; x(j) x(j+1)];
    end
end
%=====
% FIN du fichier PassageZero.m
%=====

```

```

%=====
% FICHIER : bissec.m
% DESCRIPTION : Effectue une bisection
% PARAMÈTRES :
%      a11 ... b33 : paramètres de la fonction
%      X1 et X2 : Bornes du zéro potentiel
%      iteFct : Numéro de la fonction Fi analysée
% SORTIES :
%      Xmid : Résultat de la bisection
%=====
function Xmid =
bissec(a11,a12,a13,a22,a23,a33,b11,b12,b13,b22,b23,b33,X1,X2,iteFct)
% Initialisation
XL = X1;
XR = X2;
XmidPast = inf;
Xmid = (XL+XR)/2;
% bisection
while(abs(XmidPast-Xmid)>1E-16)
    XmidPast = Xmid;
    Ymid = SystEquOMPC(a11,a12,a13,a22,a23,a33,b11,b12,...
        b13,b22,b23,b33,Xmid,iteFct);
    YL = SystEquOMPC(a11,a12,a13,a22,a23,a33,b11,b12,b13,b22,b23,b33,XL,iteFct);
    YR = SystEquOMPC(a11,a12,a13,a22,a23,a33,b11,b12,b13,b22,b23,b33,XR,iteFct);
    if(sign(Ymid) == sign(YL))
        XL = Xmid;
    else
        XR = Xmid;
    end
    Xmid = (XL+XR)/2;
end
%=====
% FIN du fichier bissec.m
%=====

```

```

%=====
% FICHIER : SystEquOMPC.m
% DESCRIPTION : Ce fichier retourne la valeur de la fonction Fi pour une
%               valeur de A1
% PARAMÈTRES :
%   a11 ... b33 : paramètres de la fonction
%   A1 : paramètre A1
%   iteFct : Numéro de la fonction Fi analysée
% SORTIES :
%   Y : valeur de la fonction Fi
%=====
function Y = SystEquOMPC(a11,a12,a13,a22,a23,a33,...
                        b11,b12,b13,b22,b23,b33,A1,iteFct)
d = (a33*b23+b22*a23-a22*b23-b33*a23);
c = (-a12*A1*b23+a33*b13*A1+b12*A1*a23-a22*b13*A1-...
      b33*a13*A1+b22*a13*A1);
b = (-a11*A1^2*b23-a12*A1^2*b13+b33*a23+a33*A1^2*b23+b12*A1^2*a13-...
      a33*b23+b11*A1^2*a23-b33*A1^2*a23);
a = -a11*A1^3*b13+b33*a13*A1-a33*b13*A1-
      b33*A1^3*a13+a33*A1^3*b13+b11*A1^3*a13;...
poly = [d c b a];
racines = roots(poly);
racines = sort(racines);
A2 = racines(iteFct);
if(A2 ~=0)
    A3 = (-a33-a11*A1^2+a33*A1^2-a12*A1*A2-
          a22*A2^2+a33*A2^2)/(a13*A1+a23*A2);...
else
    A3 = inf;
end
Y = (A3-sign(A3)*sqrt(1-A1^2-A2^2));
%=====
% FIN du fichier SystEquOMPC.m
%=====

```

```

%=====
% FICHIER : TestOrthoIntra.m
% DESCRIPTION : Ce fichier permet vérifier si un vecteur respect les
%               contraintes quadratiques d'orthogonalité
% PARAMÈTRES :
%   Vecteur : Vecteur analysé
%   pas : Pas de décallage
%   level : Niveau
%   N : Taille du support du vecteur de la plus petite échelle.
% SORTIES :
%   ProdScalMax : Max. du prod. Scal. Intra-niveau
%=====
function ProdScalMax = TestOrthoIntra(Vecteur,pas,level,N)
% Initialisation
ProdScal = [];
ContOrthoIntra = [];
% Facteur d'echelle
FactEch = 2^(level-1);
% Contraintes quadratiques d'ortho intra-niveau
for i=1:N/pas-1
    NbPtsOverlap = i*pas*2^(level-1);
    ContOrthoIntra(:,i) = [Vecteur(end-NbPtsOverlap+1:end);...
        zeros(FactEch*N-NbPtsOverlap,1)];
end
ContOrthoIntra = [Vecteur ContOrthoIntra];
NbCont = size(ContOrthoIntra,2);
% Produit scalaire entre les vecteurs
for z=2:NbCont
    ProdScal = [ProdScal; ContOrthoIntra(:,1)*ContOrthoIntra(:,z)];
end
ProdScalMax = max(abs(ProdScal));
%=====
% FIN du fichier TestOrthoIntra.m
%=====

```

```
%=====
% FICHIER : TestOrthoInter.m
% DESCRIPTION : Ce fichier permet vérifier qu'un vecteur est ortho.
%             aux vecteurs des niveaux inférieurs
% PARAMÈTRES :
%     Vecteurs : Vecteurs des petites échelles
%     pas : Pas de décallage
%     level : Niveau
%     N : Taille du support du vecteur de la plus petite échelle.
% SORTIES :
%     ProdScalMax : Max. du produit scalaire entre un vecteur et ceux
%                 des niveaux inférieurs
%=====
```

```
function ProdScalMax = TestOrthoInter(Vecteurs,pas,level,N)

% Initialisation
ProdScal = [];
% Facteur d'echelle
FactEch = 2^(level-1);
ContOrthoTot = Vecteurs{level};
% Vecteurs de petites échelles
for i=1:level-1
    ContOrthoNivInf = DecalVecteur(Vecteurs{i},FactEch*N,2^(i-1)*pas);
    ContOrthoTot = [ContOrthoTot ContOrthoNivInf];
end
NbCont = size(ContOrthoTot,2);
for z=2:NbCont
    ProdScal = [ProdScal; ContOrthoTot(:,1)'*ContOrthoTot(:,z)];
end
ProdScalMax = abs(max(ProdScal));
%=====
% FIN du fichier TestOrthoInter.m
%=====
```

```

%=====
% FICHIER : VerifBaseComplete.m
% DESCRIPTION : Ce fichier permet de vérifier qu'une base de DOMPC est
%               orthonormale
% PARAMÈTRES :
%               BaseComplete : ensemble des vecteurs de la base DOMPC
% SORTIES :
% aucune
%=====
function VerifBaseComplete(BaseComplete)

disp('-----')
disp('Debut de la verification de l"orthonormalite de la base')
ProdScal = [];
for i=1:size(BaseComplete,2)-1
    norme(i) = norm(BaseComplete(:,i));

    for j=i+1:size(BaseComplete,2)
        ProdScal = [ProdScal BaseComplete(:,i)*BaseComplete(:,j)];
    end
end
disp(['Norme min: ' num2str(min(norme))])
disp(['Norme max: ' num2str(max(norme))])
disp(['Prod. Scal. max: ' num2str(max(abs(ProdScal)))]])
disp('Verification de l"orthonormalite terminee')
%=====
% FIN du fichier VerifBaseComplete.m
%=====

```