

---

# Kernel Basis Pursuit

Vincent Guigue – Alain Rakotomamonjy – Stéphane Canu

Lab. d'Informatique, de Traitement de l'Information et des Systèmes (LITIS) EA 4051  
Avenue de l'Université, F-76801 St Étienne du Rouvray

---

*ABSTRACT.* Estimating a non-uniformly sampled function from a set of learning points is a classical regression problem. Kernel methods have been widely used in this context, but every problem leads to two major tasks: optimizing the kernel and setting the fitness-regularization compromise. This article presents a new method to estimate a function from noisy learning points in the context of RKHS (Reproducing Kernel Hilbert Space). We introduce the Kernel Basis Pursuit algorithm, which enables us to build a  $\mathcal{L}_1$ -regularized-multiple-kernel estimator. The general idea is to decompose the function to learn on a sparse-optimal set of spanning functions. Our implementation relies on the Least Absolute Shrinkage and Selection Operator (LASSO) formulation and on the Least Angle Regression Stepwise (LARS) solver. The computation of the full regularization path, through the LARS, will enable us to propose new adaptive criteria to find an optimal fitness-regularization compromise. Finally, we aim at proposing a fast parameter-free method to estimate non-uniform-sampled functions.

*RÉSUMÉ.* Estimer une fonction échantillonnée irrégulièrement à partir d'un ensemble de points constitue un problème de régression classique. Des solutions basées sur les méthodes à noyaux existent mais leur mise en œuvre conduit à deux problèmes récurrents de sélection de modèles : l'optimisation des paramètres du noyau et le réglage du compromis biais-variance. Cet article présente une méthode novatrice pour estimer une fonction à partir d'un ensemble de points bruités dans le contexte des espaces de Hilbert à noyau reproduisant. Nous avons conçu l'algorithme du Kernel Basis Pursuit pour construire une solution reposant sur des noyaux multiples et une régularisation  $\mathcal{L}_1$ . Notre idée est de décomposer la fonction à apprendre dans l'espace engendré par un dictionnaire de fonctions explicatives, à la manière des stratégies de poursuite. La mise en œuvre repose sur la formulation du LASSO (Least Absolute Shrinkage and Selection Operator) et nous avons utilisé l'algorithme Least Angle Regression Stepwise pour la résolution. Le calcul du chemin complet de régularisation nous permet d'utiliser des nouveaux critères pour déterminer automatiquement le compromis biais-variance optimal.

*KEYWORDS:* regression, multiple kernels, LASSO, parameter free.

*MOTS-CLÉS :* régression, noyaux multiples, LASSO, méthode sans paramètre.

---

## 1. Introduction

The context of our work is the following: we wish to estimate the functional dependency between an input  $x$  and an output  $y$  of a system given a set of examples  $\{(x_i, y_i), x_i \in \mathbb{R}^d, y_i \in \mathbb{R}, i = 1 \dots n\}$  which have been drawn i.i.d from an unknown probability distribution  $P(X, Y)$ <sup>1</sup>. Thus, our aim is to recover the function  $\hat{f}$  belonging to a hypothesis space  $\mathcal{H}$  which minimizes the following risk:

$$R[f] = \mathbb{E}\{(f(X) - Y)^2\} \quad [1]$$

but as  $P(X, Y)$  is unknown, we have to look for the function  $\hat{f}$  which minimizes the empirical risk:

$$R_{emp}[f] = \frac{1}{n} \sum_{i=1}^n (f(x_i) - y_i)^2 \quad [2]$$

Depending on  $\mathcal{H}$ , this problem can be ill-posed and a classical way to turn it into a well-posed one is to use regularization theory (Tikhonov *et al.*, 1977; Girosi *et al.*, 1995). In this framework, the solution of the problem is the function  $\hat{f} \in \mathcal{H}$  that minimizes the regularized empirical risk:

$$R_{reg}[f] = \frac{1}{n} \sum_{i=1}^n (y_i - f(x_i))^2 + \lambda \Omega(\|f\|_{\mathcal{H}}) \quad [3]$$

where  $\Omega$  is a functional which measures the smoothness of  $\hat{f}$  and  $\lambda$  a regularization parameter (Wahba, 1990). Under general conditions on  $\mathcal{H}$  (Reproducing Kernel Hilbert Space) (Kimeldorf *et al.*, 1971), the solution of this minimization problem can be written as:

$$\hat{f}(x) = \sum_{i=1}^n \beta_i K(x_i, x) \quad [4]$$

where  $K$  is the reproducing kernel of  $\mathcal{H}$ .

The objective of the Kernel Basis Pursuit (KBP) is two-fold: building a sparse multi-kernel-based solution for this regression problem and introducing new solutions for the bias-variance compromise problem. The multiple kernel approach has two advantages: it allows us to build adapted solutions for multiscale problems and it leads to an easier setting of the kernel hyperparameters. In this multiple kernel setting, the set of functions  $K_j(x_i, x)$  can be seen as a dictionary of spanning functions  $\mathcal{D}$  and the KBP solution will be a sparse decomposition of the function to be estimated based on this family of functions. The question of sparsity is addressed by using the Least Absolute Shrinkage and Selection Operator (LASSO) formulation (Tibshirani, 1996), namely using  $\Omega = \|\beta\|_1$  as a regularization term in equation (3). Using the Stepwise Least Angle Regression (LARS)(Efron *et al.*, 2004) as a solver of optimization

---

1. When  $x \in \mathbb{R}$  we estimate a non-uniformly sampled signal.

problem (3) enables us to compute the full set of regression solutions with varying  $\lambda$ . This set of optimal solutions is the so-called regularization path (Bach *et al.*, 2004). We use this property to introduce some heuristics which sets the bias-variance compromise dynamically. Combining a forward-iterative solver (LARS) with efficient early-stopping heuristics makes the KBP both sparse and fast.

The paper is organized as follows: in section 2, we will compare two common strategies to face the problem of building a sparse regression function  $\hat{f}$ : the Matching Pursuit and the Basis Pursuit. We will explain the building and the use of the multiple kernels, combined with the LARS in section 3. Section 4 deals with the setting of the bias-variance compromise and the kernel parameters. Our results on synthetic and real data are presented in section 5. Section 6 gives our conclusions and perspectives on this work.

## 2. General approach: Basis vs Matching Pursuit

The question of the sparsity of the solution  $\hat{f}$  can be addressed in two different ways. The first approach is based on stepwise method consisting in adding functions from a dictionary whereas the second one is to use a regularization term in equation (3) that imposes sparsity of  $\beta$  coefficients.

Mallat and Zhang introduced the Matching Pursuit algorithm (Mallat *et al.*, 1993): they proposed to construct a regression function  $\hat{f}$  as a linear combination of elementary functions  $g_i$  picked from a finite redundant dictionary  $\mathcal{D} = \{g_i\}$ . This algorithm is iterative and one new function  $g_i$  is introduced at each step, associated with a weight  $\beta_i$ . At step  $k$ , we get the following approximation of  $f$ :  $\hat{f}^{(k)} = \sum_{i=1}^k \beta_i g_i$ . Given  $R^{(k)}$ , the residue generated by  $\hat{f}^{(k)}$ , the function  $g_{k+1}$  and its associated weight  $\beta_{k+1}$  are selected according to:

$$(g_{k+1}, \beta_{k+1}) = \operatorname{argmin}_{g_i \in \mathcal{D}, \beta \in \mathbb{R}} \|R^{(k)} - \hat{f}^{(k)}\|^2 \quad [5]$$

The improvements described by Pati *et al.* (Orthogonal Matching Pursuit algorithm) (Pati *et al.*, 1993) keep the same framework, but optimize all the weights  $\beta$  at each step. A third algorithm called pre-fitting (Vincent *et al.*, 2002) enables us to choose  $(g_{k+1}, \beta_{k+1})$  according to  $R^{(k+1)}$ . All those methods are iterative and greedy. The different variations improve the weights or the choice of the function  $g_{k+1}$  but the main characteristic remains unchanged. Matching Pursuit does not allow to get rid of a previously selected function  $g_i$ , which means that its solution is sub-optimal.

The Basis Pursuit approach proposed by Chen *et al.* (Chen *et al.*, 1998) is different: they consider the whole dictionary of functions and look for the best linear solution to estimate  $f$ , namely, the solution which minimizes the regularized empirical risk using  $\Omega = \|\beta\|_1$ . This leads to the so-called LASSO formulation. Such a formulation requires costly and complex linear programming (Chen, 1995) or modified EM implementation (Grandvalet, 1998) to be solved. Finally it enables them to find an exact solution to the regularized learning problem.

The Stepwise Least Angle Regression (LARS) offers new opportunities, by combining an iterative and efficient approach with the exact solution of the LASSO. The fact that the LARS begins with an empty set of variables, combined with the sparsity of the solution explains the efficiency of such method. The ability of deleting dynamically useless variables enables the method to converge to the exact solution of the LASSO problem.

### 3. Learning with multiple kernels and $\mathcal{L}_1$ regularization

#### 3.1. LASSO

The Least Absolute Shrinkage and Selection Operator (LASSO) formulation (Tibshirani, 1996) relies on the combination of a  $\mathcal{L}_2$  loss function (we will use least squares) and  $\mathcal{L}_1$  regularization (we will penalize the sum of the absolute coefficients of the estimator). The LASSO is often written as a Tikhonov regularized problem (equation (3)) with  $\Omega = \|\beta\|_1$ . But LASSO can also be written as:

$$\min_{\beta} \sum_{i=1}^n (y_i - x_i^T \beta)^2 \quad [6]$$

$$\sum_{i=1}^d |\beta_i| \leq t$$

where  $t$  becomes the regularization parameter. The algorithm Stepwise Least Angle Regression (LARS) uses this last formulation.

#### 3.2. LARS: linear case

Given  $x_i \in \mathbb{R}^d$  the vectorial representation of a learning point, we build the matrix of the learning points:  $X = \begin{pmatrix} x_1^T \\ \vdots \\ x_n^T \end{pmatrix} \in \mathbb{R}^{n \times d}$ . Each column  $i$  of the matrix  $X$  is a variable denoted by  $X_i$ . The LARS (Efron *et al.*, 2004) is a stepwise iterative algorithm which provides an exact solution to the LASSO (section 3.1). When the variables  $X_i$  are normalized (standard deviation equals one), LARS turns this learning problem into a variable selection problem.

##### 3.2.1. Algorithm

Alg. 3.1 sums up LARS algorithm. We note  $\mathcal{A}$  the set of indexes of the active variables and  $X_{\mathcal{A}}$  the learning set reduced to the variables that are in  $\mathcal{A}$ . At each step  $k$ , given the residue  $R^{(k)} = y - \hat{f}^{(k)}$ , the LARS selects the variable which is most correlated with  $R^{(k)}$  and add its index to  $\mathcal{A}$ . The ability of dynamically suppressing a source of information which becomes useless enables the algorithm to fit the LASSO solution.  $\hat{f}^{(k+1)}$  belongs to the space spanned by  $X_{\mathcal{A}}$ , the  $\beta^{(k+1)}$  are computed to

minimize  $\|R^{(k+1)} - \hat{f}^{(k+1)}\|^2$  under the constraints that each variable of  $X_{\mathcal{A}}$  is equi-correlated with  $R^{(k+1)}$ . This leads to the property that each  $\hat{f}^{(k)}$  corresponds to an optimal solution of (6) for a given value of  $t$ : LARS computes the whole regularization path.

---

**Alg. 3.1** *Least Angle Regression Stepwise algorithm*

---

We note  $\mathcal{A}$  the set of indexes of the active variables and  $X_{\mathcal{A}}$  the learning set reduced to the variables that are in  $\mathcal{A}$ .

- 1) Initialize:  $\mathcal{A} = \emptyset, \bar{\mathcal{A}} = \{1, \dots, n\}, \forall i, \beta_i = 0$
  - 2) Find the variable to add in  $\mathcal{A}$  (most correlated with the residue).
  - 3) Compute  $X_{\mathcal{A}}^T X_{\mathcal{A}}$  inverse to find  $\vec{u}_{\mathcal{A}}$  so that  $f^{(k+1)} = f^{(k)} + \gamma \vec{u}_{\mathcal{A}}$ .
  - 4) Compute the step  $\gamma$  so that:  $\exists i \in \bar{\mathcal{A}}, j \in \mathcal{A}, X_i^T R^{(k+1)} = X_j^T R^{(k+1)}$ .
  - 5) Update  $\beta$  coefficients.
  - 6) Check that the signs of  $\beta$  coefficients remain the same, otherwise, reduce  $\gamma$  so that concerned  $\beta$  becomes null.
- 

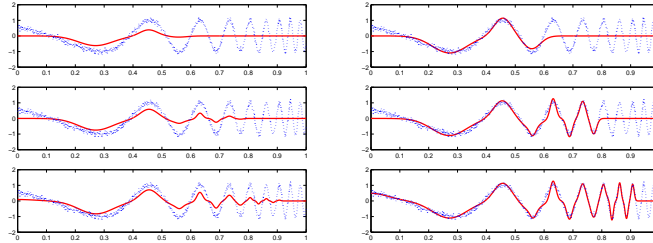
### 3.2.2. Efficiency

Solving the LASSO is really fast with this method, due to the fact that it is both forward and sparse. The first steps are not expensive, because of the small size of  $\mathcal{A}$ , then it becomes more and more time-consuming with iterations. However, owing to the  $\mathcal{L}_1$  regularization, the sparsity of the solutions limits the number of required iterations. LARS begins with an empty active set whereas other backward methods (Grandvalet, 1998; Chen, 1995) begin with all  $\beta$  being non-zero and require to solve high dimensional linear system to set irrelevant coefficients to zero. Given the fact that only one point is added (or removed) during an iteration, it is possible to update the solution at each step instead of fully computing it. This leads to a simple-LARS algorithm, similar to the simple-SVM formulation (Loosli *et al.*, 2004), which also increases the speed of the method.

### 3.2.3. Modification of the LARS

In order to improve sparsity as well as error rate, we introduce a slight modification of the LARS algorithm. We compute the step according to ordinary least square criterion in the last iteration of the method. The figure 1 illustrates this modification and the resulting improvement on an toy example.

In that case, the LARS is used as a variable selection algorithm and the resulting estimator is the partial least square solution reduced to this subset of variables. Concerning the implementation in the LARS algorithm, we only need to modify the last iteration of the algorithm and the requested time computation is not altered.  $\hat{f}^{(k+1)}$  is then designed so that the correlation  $X_{\mathcal{A}}^T R^{(k+1)}$  is null (modification of  $\gamma$  in step 4 of Alg. 3.1).



**Figure 1.** Differences between LARS regression and partial least square. The function to learn is  $\cos(\exp(\omega t))$  and the learning points are noisy. The two pictures show the results at iterations 5, 10 and 20 using the LARS (left figure) or partial least square (right figure)

### 3.3. Multiple kernels and LARS

#### 3.3.1. Building a multiple kernel regression function

In their Kernel Matching Pursuit algorithm, Vincent and Bengio (Vincent *et al.*, 2002) proposed to treat the kernel  $K$  exactly in the same way as the matrix of the learning points  $X$ . Each column of  $K$  is then considered as a variable or a source of information that can be added to the active set in order to build a linear estimation of  $f$ :  $\hat{f}(x) = \sum_{i=1}^n \beta_i K(x_i, x)$ . This function  $\hat{f}$  is  $\beta$ -linear in the Reproducing Kernel Hilbert Space (RKHS)  $\mathcal{H}$  spanned by  $K$ , and non-linear in the original data space.

To deal with the multiple kernel setting, we propose here an extension of this framework : it consists in building a set of kernel  $\{K_i\}_{i=1, \dots, N}$ , which respectively spans the spaces  $\mathcal{H}_i$ . Each source of information  $K_i(x_j, \cdot)$  is characterized by a point  $x_j$  of the learning set and a kernel parameter  $i$ . Hence, the multiple kernel  $K$  can be written as:

$$K = [ K_1 \dots K_i \dots K_N ] \quad K \in \mathbb{R}^{n \times s}, \text{ with } s = nN. \quad [7]$$

Assuming that each column of  $K$  is normalized so that its standard deviation is 1, the LARS will pick automatically the most relevant  $K_i(x_j, \cdot)$  among the whole set of sources of information<sup>2</sup>. The solution  $\hat{f}$  is a weighted sum of  $K_i(x_j, \cdot)$ :

$$\hat{f}(x) = \sum_{i=1}^N \sum_{j=1}^n \beta_{ij} K_i(x_j, x) \quad , \hat{f} \in \mathcal{H}_1 + \dots + \mathcal{H}_N \quad [8]$$

2. At each iteration, LARS selects the variable which is most correlated with the residue. This normalization makes the comparison fair.

The Kernel Basis Pursuit algorithm consists in solving the following problem:

$$\min_{\beta} \|y - K\beta\|^2 \quad [9]$$

$$\sum_{i,j} |\beta_{ij}| \leq t$$

It is important to note that no assumption is made on the kernels  $K_i$  which can be non-positive.  $K$  can associate kernels of the same type (e.g. Gaussian) with different parameter values as well as different types of kernels (e.g. Gaussian and polynomial). The resulting matrix  $K$  is neither positive definite nor square. As a consequence, any dictionary of functions could be used. The introduction of kernels enables us to build easily an efficient dictionary and to compare the resulting estimators with SVM ones.

### 3.3.2. Examples of multiple kernels

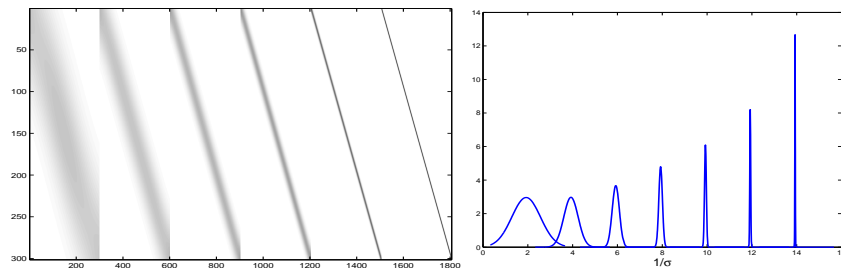
The first idea was to build a series of kernels  $K_i$  based on popular kernel formulations. We implemented some polynomial kernels  $K_i$ , based on a series of power parameters  $p_i \in \mathbb{N}$ :

$$K_i(x, x') = (\langle x, x' \rangle + 1)^{p_i} \quad [10]$$

and some Gaussian kernels, based on a series of Gaussian parameters  $\sigma_i$ :

$$K_i(x, x') = \exp\left(-\frac{\|x - x'\|^2}{2\sigma_i^2}\right) \quad [11]$$

The resulting matrix and some functions corresponding to different bandwidths are illustrated on figure 2. We will see further that the large-bandwidth functions will be used to reconstruct the low frequency regions and the small-bandwidth-functions will be used to reconstruct the high frequency regions of the original signal.



**Figure 2.** Multiple Gaussian kernel matrix. This matrix is obtained by concatenation of different Gaussian kernels with decreasing bandwidths. The corresponding functions will enable us to reconstruct different parts of the original signal

Another approach to build a multiple kernels relies on multiscale analysis. The general idea is to build a kernel from a series a analytic functions. (Rakotomamonjy *et al.*, 2005) proposed a method to perform this kind of operation to build a reproducing kernel, they use any family of functions  $\{\psi_n\}_{i=1\dots n}$  satisfying:

- any set of functions  $\{\psi_n\}_{i=1\dots n}$  of  $\mathcal{H}$  is a frame of the spanned subspace (Mallat, 1997),
- if each  $\psi_n$  of the set is bounded and its norm is finite in  $\mathcal{H}$ , then the spanned space is a RKHS.

In that case, the kernel of this RKHS can be written as:

$$K(x, x') = \sum_{i=1}^n \bar{\psi}_i(x) \psi_i(x') \quad [12]$$

where  $\bar{\psi}_i$  is the dual frame of  $\psi_i$ . Finally, a finite family of wavelet functions always generates a RKHS.

The figure 3 shows the building of such a kernel. Given a wavelet family of functions  $\{\psi_{jk}\}$  where index  $j$  indicates the scale and index  $k$  indicates the time location of the analytic function, we compute the kernel matrix  $K_{jk}$  such as:

$$K_{jk}(x, x') = \bar{\psi}_{jk}(x) \psi_{jk}(x') \quad [13]$$

We perform this operation for all the wavelet functions. Then we build the kernel matrix  $K_j$  according to (12):

$$K_j = \sum_k K_{jk} \quad [14]$$

$K_j$  is the sum of all the translations of a given mother wavelet  $\psi_j$ , for a given scale  $j$ . Finally, each kernel  $K_j$  is related to one scale and each source of information can be written as:

$$K_j(x, \cdot) = \sum_k \bar{\psi}_{jk}(x) \psi_{jk}(\cdot) \quad [15]$$

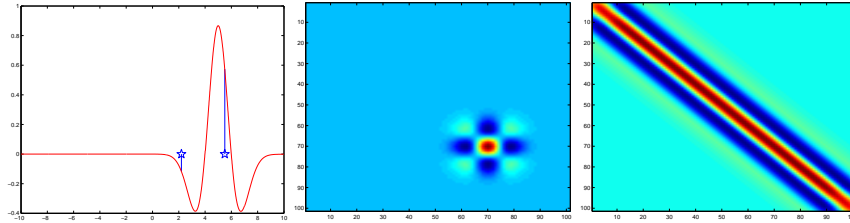
As shown on figure 3, the rank of  $K_{jk}$  is 1 whereas the rank of  $K_j$  is  $n$ .

#### 4. Setting of regularization and kernel parameters

The optimization problem (9) requires the setting of several hyperparameters: the bias-variance compromise  $t$  as well as the  $N$  kernel hyperparameters. Considering  $N$  kernels with a single parameter, this would lead to  $N+1$  parameters selection problem. It can be solved by cross-validation but it is very expensive in time-computation. In the following section, we propose some strategies to tackle this problem.

##### 4.1. Optimization of regularization parameter

Finding a good setting for  $t$  in equation (9) is very important: when  $t$  becomes too large, the LARS becomes equivalent to Ordinary Least Square (OLS) and it requires



**Figure 3.** Building of  $K_j$  based on wavelets. The left picture shows the computation of a point in the kernel matrix  $K_{jk}(x, x') = \bar{\psi}_{jk}(x)\psi_{jk}(x')$ , the learning points  $x$  and  $x'$  are drawn as stars. The second picture presents the kernel matrix for a given function  $\psi_{jk}$ , each line is related to a value of  $x$  and each column is related to a value of  $x'$ . The right picture is the sum of many kernels, corresponding to all the translations of a given wavelet  $\psi_j$

the resolution of linear system of size  $s \times s$ . Early stopping enables us to decrease the time computation (which is linked to the sparsity of the solution) as well as to improve the generalization of the learning (by regularizing).

One of the most interesting property of the LARS is the fact that it computes the whole regularization path (section 3.2). The LARS enables us to compute a set of optimal solutions corresponding to different values of  $t$ , with only one learning stage. We are going to take advantage of this property to select the optimal  $t$  dynamically.

#### 4.1.1. Different compromise parameters

The original formulation of the LARS relies on the compromise parameter  $t$  which is a bound on the sum of the absolute values of the  $\beta$  coefficients.  $t$  is difficult to set because it is somewhat meaningless. We look for different expressions of the regularization parameter  $t$ . The aim is to find the most meaningful one, namely the easiest way to set this parameter.

- LC-KBP. It is possible to apply Ljung Criterion (Ljung, 1987) on the autocorrelation of the residue. The parameter is then a threshold which decides when the residue can be considered as white noise.

- $\nu$ -KBP. Another solution consists in defining a criterion on the size of  $\mathcal{A}$ , namely on the number of support vectors or on the rate of support vectors among the learning set. It is important to note that  $\nu$  is then a threshold, whereas in the  $\nu$ -SVM method where  $\nu$  can be seen as an upper bound on the rate of support vectors (Schölkopf *et al.*, 2002).

However, all these methods require the *a priori* setting of a parameter which is usually estimated by cross-validation.

#### 4.1.2. Trap source

We propose a novel method to select dynamically the regularization parameter based on a trap parameter. The idea is to introduce one or many sources of information that we do not want to use. When the most correlated source with the residue belongs to the trap set, the learning procedure is stopped. We use two heuristics to build the trap sources:

- KBP- $\sigma_s$ : we build a Gaussian kernel  $K_{\sigma_s}(x_i, x_j) = \exp\left(-\frac{\|x_i - x_j\|^2}{2\sigma_s^2}\right)$  and we add it to the information sources.  $\sigma_s$  is a very small bandwidth.
- KBP-RV: we add iid Gaussian random variables among the sources of information. This heuristic has already been used successfully in variable selection methods (Bi *et al.*, 2003).

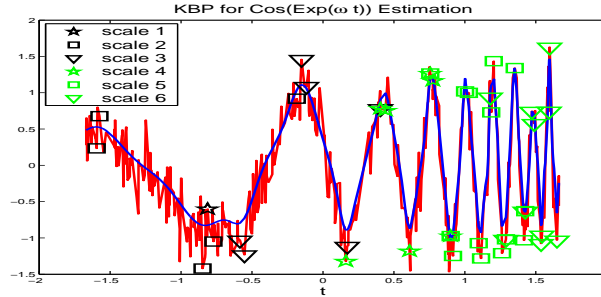
The use of a trap scale is closely linked to the way that LARS selects the sources of information (section 3.2). We illustrate the behavior of the trap scale during the learning of a toy function:  $\cos(\exp(\omega x))$  (figure 4(a)). The correlation with the residue is an energetic criterion, that is why the first selected variables explain the low frequency regions of the  $\cos(\exp(\omega x))$  function. The selected sources of information belong to higher and higher scales with iterations (figure 4(c)). If we further assume that  $y$  contains white noise, then there is no correlation between noises occurring at two different instants: the noise is a local phenomenon. As a consequence, the sources of information that explain the noise will belong to narrow bandwidth Gaussian kernel  $K_{\sigma_s}$  and will be selected at the end of the learning procedure. Moreover, the selection of a source from  $K_{\sigma_s}$  means that there are no more correlated sources of information in other  $K_i$ , namely the residue is only composed of components correlated with noise.

The use of Gaussian random variables as a trap scale is more intuitive: it supposes that when a random variable is selected as the most correlated source with the residue, it remains no more interesting information in the residue. Figure 4(b) illustrates the evolution of the residue with iterations, the 3<sup>rd</sup> picture shows the residue when a random variable is selected.

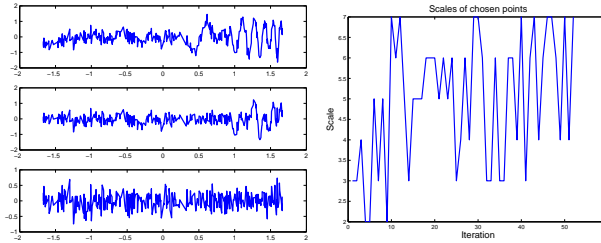
#### 4.2. Optimizing Gaussian kernel parameters

In this section, we propose a new method to set the kernel parameters of the KBP in the Gaussian case, without using cross-validation. We aim at finding a key parameter  $\sigma_k$  representing the smallest bandwidth which can be useful for a given problem. In this case:  $K_{\sigma_k} \approx I_n$  and  $\hat{f}$  will interpolate  $y$  and overfit. Then, we propose to build a series of larger and larger Gaussian parameters from this key scale to improve the generalization of the learning.

$\sigma_k$  is obtained according to the following steps: a one nearest neighbor is performed on the training data. Then, we focus on the shortest distances between neigh-



(a) Learning of the  $\cos(\exp(\omega x))$  function.



(b) Residues at iteration 10, 20 (c) Scale of the variable selected at each step of the LARS.

**Figure 4.** Evolution of the solution with iterations

bors. The key distance  $D_k$  is the distance between  $x_i$  and  $x_j$ , the two nearest points in the input space. The corresponding key Gaussian parameter  $\sigma_k$  is defined so that:

$$K_{\sigma_k}(x_i, x_j) = \exp\left(-\frac{D_k^2}{2\sigma_k^2}\right) = 0.1 \tag{16}$$

that is to say, the bandwidth  $\sigma_k$  is designed so that a learning point in high density regions has little influence on its neighbors. For more robustness, it is recommended to use an improved definition of  $D_k$ . Given  $\mathcal{S}$  the set of the one-nearest-neighbor distances. We define  $D_k$  as the mean distance of the 0.01 quantile of  $\mathcal{S}$ .

Then, a series of bandwidth is build as follow (figure 2):

$$\sigma = \{\sigma_k, \sigma_k p, \sigma_k p^2, \sigma_k p^3, \sigma_k p^4, \sigma_k p^5, \sigma_k p^6\} \quad \text{with: } p > 1 \tag{17}$$

A small value of  $p$  provides more accuracy for the design of the sources of information, when  $p$  becomes close to 1, the family becomes exhaustive. However, a small value of  $p$  leads to a more redundant family of functions, which penalizes the sparsity of the solution. In fact, when two sources of information are correlated, they are both almost equi-correlated to the residue and the LARS often selects both sources of information in two successive iterations. The 2<sup>nd</sup> line of table 1 shows that the number of support

vectors first decreases with  $p$  due to this phenomenon. On the contrary, when  $p$  is too large, the accuracy of the sources of information decreases and the LARS requires many sources of information to describe a single region of the signal. That is why in our example the size of the optimal active set  $\mathcal{A}$  increases with  $p$  when  $p > 2$ .

$p$	1.2	1.5	2	2.5	3	4	5	6
MSE	0.0225	0.0222	0.0223	0.0219	0.0209	<b>0.0205</b>	0.0249	0.0254
$ \mathcal{A} $	45.33	44.66	<b>39.0</b>	40.25	44.5	48.33	48.66	48.33

**Table 1.** Estimation of the function  $\cos(\exp(\omega x))$  (10 runs). Evolution of the best solution in terms of sparsity (means of used support vectors over 10 runs) and Mean Square Error (MSE) in function of parameter  $p$

Cross-validations over synthetic and real data lead to set  $p = 3$ . We choose to set the cardinality of  $\sigma$  to 6, given the fact that experimental results are not improved beyond this value.

KBP results presented in the next section rely on this parameter-free Gaussian strategy, but it is also possible to build multiple kernels with different degrees of polynomial kernels or to mix different kernels.

## 5. Experiments

We illustrate the efficiency of the methodology on synthetic and real data. Tables 2, 3 and 4 present the results with two different algorithms: the SVM and the KBP. We use two kinds of kernels (Gaussian and wavelet) and four strategies to stop the learning stage of the KBP. All results use the modification of LARS presented in section 3.2.3.

– KBP- $\sum_i |\beta_i|$  is the classical method where a bound is defined on the sum of the regression coefficients. This bound is estimated by cross-validation.

–  $\nu$ -KBP is based on the fraction of support vectors.  $\nu$  is also estimated by cross-validation.

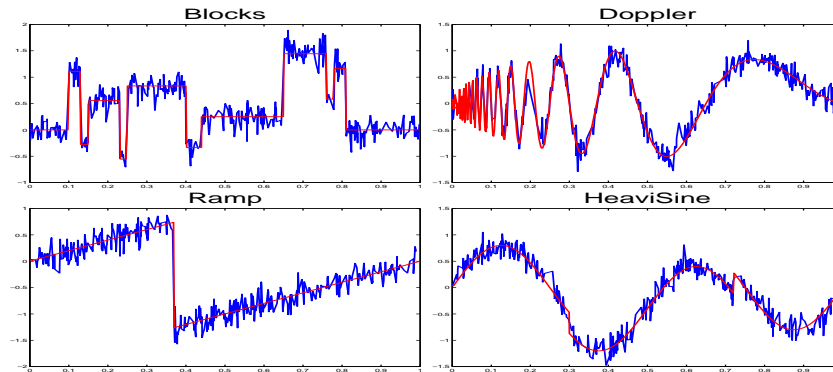
– KBP-RV relies on the introduction of Gaussian random variables as sources of information.<sup>1</sup>

– KBP- $\sigma_s$  relies on a Gaussian trap scale with very small bandwidth. We use  $\sigma_s = \sigma_k$  of equation (16).<sup>3</sup> In the wavelet kernel case, we swap from KBP- $\sigma_s$  to KBP-HF (High Frequency) and we use the kernel corresponding to the smallest scale as trap scale.

To validate this approach, we compare the results with classical Gaussian  $\epsilon$ -SVM regression. Parameters  $\epsilon$ ,  $C$  and  $\sigma$  are optimized by cross validation. In order to distinguish the benefits of the early stopping methods from the benefits of the multiple

3. To make KBP- $\sigma_s$  and KBP-RV methods more robust, we wait until 3 information sources from the trap-scale are selected to stop the learning stage.

kernel learning, we also give the results of KBP algorithm when using a single kernel<sup>4</sup>. In this case, the kernel is chosen by cross-validation. KBP-RV and KBP- $\sigma_s$  are fully parameter-free. We use the method described in the previous section 4.2 to build our multiple kernels, with  $p = 3$ .



**Figure 5.** Toy signals and learning data

### 5.1. Synthetic data

We test our method for the learning of  $\cos(\exp(\omega x))$  regression function. We try to learn:

$$f(x) = \cos(\exp(\omega x)) + b(x) \quad [18]$$

where  $b(x)$  is a Gaussian white noise of variance  $\sigma_b^2 = 0.15$ . We also tested the method over classical synthetic data described by Donoho and Johnstone (Donoho *et al.*, 1994).

For all signals, we use  $x \in [0, 1]$ , drawn according to a uniform distribution. We use 400 points for the learning set and 1000 points for the testing set. The noise is added only on the learning set. Parameters ( $\nu, \sum_i |\beta_i| \dots$ ) are computed by cross validation on the learning set. Table 2 presents the results over 30 runs for each dataset.

These results point out the sparsity and the efficiency of KBP solutions both with Gaussian or wavelet kernels. Figure 4(a) illustrates how multiple kernels learning enables the regression function to fit the local frequency of the model. The results with different Donoho's synthetic signals enable us to distinguish the benefits of the KBP method from the benefits of the multiple kernels. The KBP improves the sparsity of the solution, whereas the multiple kernels improve the results on signals that require a multi-scale approach.

4. Even with the single kernel case, the KBP relies on a multiple kernel architecture to add the trap scales.

Nb kernel	<b>1</b>				
Algorithm	$\epsilon$ - SVM	KBP- $\sum_i  \beta_i $	$\nu$ -KBP	KBP-RV	KBP- $\sigma_s$
$\cos(e^{\omega t})$	0.032 $\pm$ 0.0086 105.4 0	0.028 $\pm$ 0.0051 87.3 0	0.028 $\pm$ 0.0052 85 0	0.026 $\pm$ 0.0047 95.3 0	0.029 $\pm$ 0.0062 95.3 0
Doppler	0.023 $\pm$ 0.0071 59.3 0	0.019 $\pm$ 0.0068 47.1 0	0.019 $\pm$ 0.0060 47 0	0.020 $\pm$ 0.0053 48.8 0	0.019 $\pm$ 0.0056 51.3 0
Blocks	0.039 $\pm$ 0.0020 75.3 0	0.025 $\pm$ 0.0013 65.1 0	0.026 $\pm$ 0.0015 65 0	0.024 $\pm$ 0.0012 69.4 0	0.024 $\pm$ 0.0012 65.2 0
Ramp	<b>0.0072 <math>\pm</math> 0.0034</b> 54.2 <b>16</b>	0.0114 $\pm$ 0.0013 45.6 0	0.0112 $\pm$ 0.0022 48 0	0.0107 $\pm$ 0.0020 45.9 0	0.0108 $\pm$ 0.0015 49.7 0
HeaviSine	<b>0.0028 <math>\pm</math> 0.0002</b> 51.4 11	<b>0.0028 <math>\pm</math> 0.0002</b> 17.2 <b>15</b>	0.0030 $\pm$ 0.0002 20 5	0.0029 $\pm$ 0.0002 21.1 8	0.0028 $\pm$ 0.0002 19.5 11

Nb kernel	<b>6 (Multiple Kernels)</b>			
Algorithm	KBP- $\sum_i  \beta_i $	$\nu$ -KBP	KBP-RV	KBP- $\sigma_s$
$\cos(\exp(t))$	<b>0.020 <math>\pm</math> 0.0039</b> 47.4 14	0.023 $\pm$ 0.0059 46 5	<b>0.020 <math>\pm</math> 0.0039</b> 48.4 <b>15</b>	0.021 $\pm$ 0.0035 47.8 10
Doppler	0.011 $\pm$ 0.0052 46.1 13	0.013 $\pm$ 0.0060 46 3	<b>0.010 <math>\pm</math> 0.0059</b> 52.70 <b>17</b>	0.013 $\pm$ 0.0055 49.80 5
Blocks	0.020 $\pm$ 0.0011 64.6 13	0.020 $\pm$ 0.0012 65 10	0.020 $\pm$ 0.0012 67.5 9	<b>0.019 <math>\pm</math> 0.0011</b> 66.3 <b>17</b>
Ramp	<b>0.0072 <math>\pm</math> 0.0033</b> 20.1 12	0.0073 $\pm$ 0.0035 18 11	0.0080 $\pm$ 0.0030 22.5 5	0.0077 $\pm$ 0.0029 20.3 4
HeaviSine	0.0035 $\pm$ 0.0003 44.30 0	0.0036 $\pm$ 0.0003 45 0	0.0032 $\pm$ 0.0003 48.2 0	0.0032 $\pm$ 0.0003 49.0 0

**Table 2.** Results of SVM and KBP for the estimation of  $\cos(\exp(x))$  and Donoho's classical functions with single and multiple Gaussian kernels. Means and standard deviations of MSE on the test set (30 runs), number of support vectors used for each solution, number of best performances. The total of best performances sometimes exceeds 30 due to the fact the 2 different KBP can lead to the same solution

$\epsilon$ -SVM achieves the best results for Ramp and HeaviSine signals. This can be explained by the fact that the Ramp and HeaviSine signals are almost uniform in term of frequency. The  $\epsilon$  tube algorithm of the SVM regression is especially efficient on this kind of problem.

The results from the different KBP are very close (and often similar), that is why the total of best performances sometimes exceeds 30 (the number of run). Then, KBP-RV and KBP- $\sigma_s$  or KBP-HF become very attractive, due to the fact that they are parameter-free methods. KBP obtains the best results for 4 experiments, and parameter-free-KBP for 3 experiments on a total of 5 experiments.

Algorithm	$\epsilon$ -SVM	KBP- $\sum_i  \beta_i $	$\nu$ -KBP	KBP-RV	KBP-HF
$\cos(e^{\omega t})$	$0.029 \pm 0.0056$ 111.1 0	<b><math>0.020 \pm 0.0047</math></b> 42.3 24	$0.020 \pm 0.0048$ 42 19	$0.021 \pm 0.0049$ 48.7 16	<b><math>0.020 \pm 0.0047</math></b> 43.4 20
Doppler	$0.025 \pm 0.0056$ 110.7 0	$0.014 \pm 0.0042$ 44.1 11	<b><math>0.013 \pm 0.0037</math></b> 42 24	$0.014 \pm 0.0045$ 47.1 9	$0.013 \pm 0.0038$ 42.5 21
Blocks	$0.026 \pm 0.0045$ 107.7 0	$0.021 \pm 0.0033$ 50.3 17	<b><math>0.020 \pm 0.0034</math></b> 54 21	$0.023 \pm 0.0039$ 60.8 5	$0.022 \pm 0.0036$ 49.2 8
Ramp	$0.014 \pm 0.0037$ 57.6 2	<b><math>0.008 \pm 0.0035</math></b> 19.2 19	<b><math>0.008 \pm 0.0035</math></b> 17 20	$0.009 \pm 0.0037$ 21.4 13	$0.010 \pm 0.0035$ 15.8 8
HeavySine	<b><math>0.0028 \pm 0.0005</math></b> 29 28	$0.0034 \pm 0.0008$ 17.4 0	$0.0033 \pm 0.0008$ 18 2	$0.0033 \pm 0.0007$ 19.6 0	$0.0034 \pm 0.0008$ 20.7 2

**Table 3.** Results of SVM and KBP for the estimation of  $\cos(\exp(x))$  and Donoho's classical functions with multiple wavelet kernels. Means and standard deviations of MSE on the test set (30 runs), number of support vectors used for each solution, number of best performances. The total of best performances sometimes exceeds 30 due to the fact the 2 different KBP can lead to the same solution

## 5.2. Real data

Experiments are carried out over regression data bases available in the UCI repository (Blake *et al.*, 1998). We compare our results with (Chang *et al.*, 2005).

The experimental procedure for real data is the following one: Thirty training/testing sets are randomly produced. Respectively 80% and 20% of the points are used for training and testing. Hyperparameters ( $\nu$ ,  $\sum_i |\beta_i| \dots$ ) are computed by cross-validation on the learning set. Table 4 presents means and standard deviations of MSE (mean square error) on the test set.

$\epsilon$ -SVM solution is not really competitive but it gives an interesting information on the number of support vectors required for each solution. KBP-RV and KBP- $\sigma_s$  results are very interesting: they are parameter free using the heuristic described in section 4.2, moreover the KBP-RV achieves the best result for pyrim.

## 6. Conclusion

The Kernel Basis Pursuit algorithm enables us to meet two objectives: proposing a sparse multi-kernel-based solution for the regression problem and introducing new solutions for the bias-variance compromise problem and the kernel setting.

The sparsity is due to  $\mathcal{L}_1$  regularization, and the interpretation of the  $K_i(x_j, \cdot)$  as simple source of information enables the KBP to deal with multiple kernels. The heuristics proposed to set the different parameters or compromises of the KBP rely both on the LARS properties and the multiple kernel: multiple kernels allow easy

Nb kernel	<b>1</b>				
Algo	SVM		KBP		
	(Chang <i>et al.</i> , 2005)	$\epsilon$ -SVM	$\sum_i  \beta_i $	RV	$\sigma_s$
pyrim	$0.007 \pm 0.007$	$0.008 \pm 0.010$	$0.010 \pm 0.011$	$0.011 \pm 0.009$	$0.010 \pm 0.011$
	–	47.1	29.7	31.2	33.6
	–	0	0	0	0
triazines	$0.021 \pm 0.005$	$0.022 \pm 0.006$	$0.021 \pm 0.006$	$0.021 \pm 0.005$	$0.021 \pm 0.006$
	–	60.8	27.0	32.4	29.0
	–	0	0	0	0
housing	<b><math>9.19 \pm 2.73</math></b>	$17.54 \pm 4.18$	$12.83 \pm 3.31$	$14.12 \pm 3.17$	$12.33 \pm 3.04$
	–	405.4	289.0	295.2	304.2
	–	0	0	0	0
abalone	<b><math>5.071 \pm 0.678</math></b>	$12.871 \pm 0.461$	$9.293 \pm 0.518$	$9.181 \pm 0.421$	$10.311 \pm 0.513$
	–	652.2	491.9	502.3	546.2
	–	0	0	0	0

Nb kernel	<b>6 (Multiple Kernels)</b>		
Algo	KBP		
	$\sum_i  \beta_i $	RV	$\sigma_s$
pyrim	$0.006 \pm 0.006$	$0.006 \pm 0.006$	<b><math>0.005 \pm 0.006</math></b>
	47.0	48.1	49.3
	13	17	<b>18</b>
triazines	<b><math>0.019 \pm 0.006</math></b>	$0.020 \pm 0.005$	$0.020 \pm 0.008$
	23.0	27.1	27.8
	<b>22</b>	8	9
housing	$10.52 \pm 3.56$	$11.03 \pm 3.31$	$10.13 \pm 3.23$
	305.4	317.3	315.2
	12	9	<b>17</b>
abalone	$7.189 \pm 0.568$	$8.363 \pm 0.616$	$8.127 \pm 0.620$
	607.4	512.8	548.3
	<b>17</b>	7	10

**Table 4.** Results of SVM and KBP for the different regression databases. Means and standard deviations of MSE on the test set (30 runs), number of support vectors used for each solution, number of best performances. The total of best performances sometimes exceeds 30 due to the fact the 2 different KBP can lead to the same solution

and efficient setting for the kernel parameters and the fact that LARS computes the whole regularization path enables us to implement powerful early-stopping strategies. The KBP gives good results on synthetic and real data. In the meantime, the required time computation is reduced compared with SVM, due to the sparsity of the obtained solutions. Moreover, the KBP becomes fully parameter-free in the KBP-RV and KBP- $\sigma_k$  cases and though they achieve very competitive results.

The perspectives of this work are the following ones: we now plan to use this description of the data for signal classification purpose. Then, the idea would be to optimize the representation of the data for the classification task.

## Acknowledgements

This work was supported in part by the IST Program of the European Community, under the PASCAL Network of Excellence, IST-2002-506778. This publication only reflects the authors' views.

## 7. References

- Bach F., Thibaux R., Jordan M., « Computing regularization paths for learning multiple kernels », *Advances in Neural Information Processing Systems*, vol. 17, 2004.
- Bi J., Bennett K., Embrechts M., Breneman C., Song M., « Dimensionality Reduction via Sparse Support Vector Machines », *Journal of Machine Learning Research*, vol. 3, p. 1229-1243, 2003.
- Blake C., Merz C., UCI Rep. of machine learning databases, <http://www.ics.uci.edu/~mllearn/MLRepository.html>, Technical report, University of California, Irvine, Dept. of Information and Computer Sciences, 1998.
- Chang M., Lin C., « Leave-one-out Bounds for Support Vector Regression Model Selection », *Neural Computation*, 2005.
- Chen S., Basis Pursuit, PhD thesis, Department of Statistics, Stanford University, 1995.
- Chen S., Donoho D., Saunders M., « Atomic Decomposition by Basis Pursuit », *SIAM Journal on Scientific Computing*, vol. 20, n° 1, p. 33-61, 1998.
- Donoho D., Johnstone I., « Ideal spatial adaptation by wavelet shrinkage », *Biometrika*, vol. 81, p. 425-455, 1994.
- Efron B., Hastie T., Johnstone I., Tibshirani R., « Least angle regression », *Annals of statistics*, vol. 32, n° 2, p. 407-499, 2004.
- Girosi F., Jones M., Poggio T., « Regularization Theory and Neural Networks Architectures », *Neural Computation*, vol. 7, n° 2, p. 219-269, 1995.
- Grandvalet Y., « Least absolute shrinkage is equivalent to quadratic penalization », *ICANN*, p. 201-206, 1998.
- Kimeldorf G., Wahba G., « Some results on Tchebycheffian spline functions. », *J. Math. Anal. Applic.*, vol. 33, p. 82-95, 1971.
- Ljung L., *System Identification - Theory for the User*, 1987.
- Loosli G., Canu S., Vishwanathan S., Smola A. J., Chattopadhyay M., « Une boîte à outils rapide et simple pour les SVM », in , M. Liquière, , M. Sebban (eds), *CAP*, Presses Universitaires de Grenoble, p. 113-128, 2004.
- Mallat S., *A Wavelet Tour Of Signal Processing*, Academic Press, 1997.
- Mallat S., Zhang Z., « Matching pursuits with time-frequency dictionaries », *IEEE Transactions on Signal Processing*, vol. 41, n° 12, p. 3397-3415, 1993.
- Pati Y. C., Rezaifar R., Krishnaprasad P. S., « Orthogonal matching pursuits : recursive function approximation with applications to wavelet decomposition », *Proceedings of the 27th Asilomar Conference in Signals, Systems, and Computers*, 1993.
- Rakotomamonjy A., Canu S., « Frame, Reproducing Kernel, Regularization and Learning », *JMLR*, vol. 6, p. 1485-1515, 2005.

Schölkopf B., Smola A., *Learning with kernels*, MIT Press, 2002.

Tibshirani R., « Regression shrinkage and selection via the lasso », *J. Royal. Statist.*, vol. 58, n° 1, p. 267-288, 1996.

Tikhonov A., Arsénin V., *Solutions of ill-posed problems*, W.H. Winston, 1977.

Vincent P., Bengio Y., « Kernel Matching Pursuit », *Machine Learning Journal*, vol. 48, n° 1, p. 165-187, 2002.

Wahba G., *Spline Models for Observational Data*, Series in Applied Mathematics, Vol. 59, SIAM, 1990.

**ANNEXE POUR LE SERVICE FABRICATION**  
A FOURNIR PAR LES AUTEURS AVEC UN EXEMPLAIRE PAPIER  
DE LEUR ARTICLE ET LE COPYRIGHT SIGNE PAR COURRIER  
LE FICHER PDF CORRESPONDANT SERA ENVOYE PAR E-MAIL

1. ARTICLE POUR LA REVUE :

*RSTI - RIA – 20/2006. New Methods in Machine Learning*

2. AUTEURS :

*Vincent Guigue – Alain Rakotomamonjy – Stéphane Canu*

3. TITRE DE L'ARTICLE :

*Kernel Basis Pursuit*

4. TITRE ABRÉGÉ POUR LE HAUT DE PAGE MOINS DE 40 SIGNES :

*Kernel Basis Pursuit*

5. DATE DE CETTE VERSION :

*October 26, 2006*

6. COORDONNÉES DES AUTEURS :

– adresse postale :

Lab. d'Informatique, de Traitement de l'Information et des Systèmes  
(LITIS) EA 4051

Avenue de l'Université, F-76801 St Étienne du Rouvray

– téléphone : 02 32 95 97 00

– télécopie : 02 32 95 97 08

– e-mail : [Vincent.Guigue@insa-rouen.fr](mailto:Vincent.Guigue@insa-rouen.fr)

7. LOGICIEL UTILISÉ POUR LA PRÉPARATION DE CET ARTICLE :

L<sup>A</sup>T<sub>E</sub>X, avec le fichier de style `article-hermes.cls`,  
version 1.21 du 09/06/2005.

8. FORMULAIRE DE COPYRIGHT :

Retourner le formulaire de copyright signé par les auteurs, téléchargé sur :  
<http://www.revuesonline.com>

SERVICE ÉDITORIAL – HERMES-LAVOISIER  
14 rue de Provigny, F-94236 Cachan cedex  
Tél. : 01-47-40-67-67  
E-mail : [revues@lavoisier.fr](mailto:revues@lavoisier.fr)  
Serveur web : <http://www.revuesonline.com>