NEURAL NETWORK SMOOTHING IN CORRELATED TIME SERIES CONTEXT

F. Badran*, **S. Thiria***†

* CEDRIC, Conservatoire National des Arts et Métiers
292 rue Saint Martin - 75141 Paris cedex 03 (FRANCE).

† Laboratoire d'Océanographie et de Climatologie (LODYC), Université de PARIS 6, 4 Place Jussieu, T14 - 75005 Paris (FRANCE)

ABSTRACT

We present in this paper a Neural Network (NN) smoothing architecture for non-parametric estimation of the trend of a time series, observed at constant regular time intervals. The NN-smoother computes the trend in the state domain and minimizes a cost function with a regularization term. The regularization term is penalized by a parameter λ which forces the learning procedure to smooth in the time domain. We define a selecting criterion in order to select the best parameter λ. We prove that this criterion is an unbiased approximation of the Mean Squared Averaged Error when the noisy component of the time series is zero-mean, auto correlated, stationary process with the auto covariance coefficients equal to zero after a certain known order.

1. INTRODUCTION

In this paper, we focus on the problem of smoothing time series, observed at constant regular time intervals. By choosing as unit these time intervals, the observed series will be represented by $\{y_i ; i = 1,...,n\}$. From now on, we will focus on data originating from time series.

A theoretical framework, which handles a wide range of practical problems, assumes that observations can be represented in the time domain by an additive model of the form:

$$Y_i = \mu(i) + Z_i \qquad (1)$$

which expresses a decomposition of Y_i in a non-random function $\mu(i)$ and a centred random function Z_i. Equation (1) allows some physical interpretation of the underlying phenomenon: for example, in meteorology, $\mu(i)$ can represent the large scale phenomena, while Z_i represents the small scale ones.

Further assumptions are usually made on this decomposition: for example that $\mu(i)$ be "smooth" and Z_i "rough". Under these assumptions, the deterministic value $\mu(i)$ represents the trend at time i of the time series.

Smoothing thus aims at discovering the underlying phenomenon $\{\mu(i); i=1,..., n\}$, from observations $\{y_i; i = 1,...,n\}$. In addition, smoothing could also allow the use of observations to a predict future trend.

Existing methods, such as kernel smoothers or spline regression in the time domain are closely related. Spline smoothing is, in an asymptotic sense, a parametric kernel smoother [Silverman 84]. All these methods are "linear smoothers", which means that $\hat{\mu}(i)$, the estimation of the smoothed value $\mu(i)$, is a linear combination of all the observations y_j of the series of the form: $\hat{\mu}(i) = \sum_{i=1}^{n} y_j w_{ji}$ where w_{ij} depends on (i,j).

Neural Networks can be used for smoothing: for example, an MLP, trained through a modified Gradient Back Propagation, can implement spline smoothing in the time domain [Bishop 90].

In most cases, smoothing techniques generate families of functions g^{λ} depending on a parameter λ, which controls the degree of smoothing. Each of these functions is a solution to the minimization of a cost function, containing a regularization term parameterized by λ. The best approximation to $\mu()$ is obtained for an optimal choice of the smoothing parameter λ. Many optimal statistical criteria, for λ selection, have been proposed when the stochastic process Z_i can be considered as a white noise [Eubank 88]. A general framework proposed by Wahba deals with correlated data and proposes the use of generalised cross-validation criteria for choosing the degree of smoothing in the linear case (for kernels and splines). These model-selection criteria heavily rely on the linearity assumption and requires the estimation and inversion of the variance-covariance matrix V of $\{Z_i, i=1..n\}$[Diggle 89, Wahba 90]. In general the estimation of V requires a known correlation structure.

Various techniques have been proposed for parameter selection in NN: the evidence framework [McKay 92], the GPE(λ) criterion [Moody 92]. However, these criteria are very hard to compute practically in the case of the regularization term used for smoothing. In this paper, we propose a new criterion dedicated to neural network smoother, for optimizing the regularization coefficient λ. As distinct from the classical "leave one out" cross validation, the use of the proposed Smoothing Criterion (SC) requires the training of the network only once for each value of λ. The technique applies with mild assumptions about the nature of the stochastic process Z_i. It works for

stationary autocorrelated processes, with zero auto-covariance coefficients after some known order M.

The choice of the optimal parameter λ depends on the definition of the mesure of discrepancy between the unknown trend μ and the function g^{λ}. In the following we consider that this mesure is:

$$MASE(g^{\lambda}) = E\left[\frac{1}{n}\sum_{i=1}^{n}\left(\mu(i) - g^{\lambda}(i/y_1,y_2,....y_n)\right)^2\right]$$

Where E stands for the expectation with respect to all possible samples (y_1,y_2,....yn). In this paper we propose a technique to compute an unbiased estimator of $MASE(g^{\lambda})$. The use of this estimator for each value of λ allows the selection of an optimal smoothing with respect to the noise hypothesis.

Section 2 presents the Neural Network required for Smoothing. Section 3 is dedicated to the Smoothing Criterion (SC) which allows the selection of the adequate smoother. This smoother can deal with correlated data without estimating the variance-covariance matrix. In section 4 we discuss this method. The validity of the SC criterion is demonstrated in the last section through simulations.

2 NEURAL SMOOTHING

A classical smoothing method is to choose from the family of twice continuously differentiable functions S, such that $\int_{-\infty}^{+\infty} S''(t)^2 dt$ exists, the function S which minimizes the measure of roughness:

$$C(\lambda) = 1/2 \sum_{i=1}^{n} [\, y_i - S(i)\,]^2 \;) + \lambda \int_{-\infty}^{+\infty} S''(t)^2\, dt \qquad (2)$$

The weighting parameter λ controls the amount of smoothing, so that increasing λ leads to a smoother estimate. Theoretical results are available on such minimization and show that, for a given λ, the function S^{λ} minimizing $C(\lambda)$ is a cubic spline [Reinsch 67; Eubank 88]. In the same way, splines can be implemented by an MLP using $C(\lambda)$ as a cost function for the backpropagation. Such an adaptation of the classical backpropagation algorithm has been made by Bishop [Bishop 1990]. At the end of the learning phase, the MLP provides an approximation of the cubic spline S^{λ}. In this approach the NN architecture used for the minimization of (2) does not take into

account a temporal context of the time series. Using such a context adds more information to the problem and is helpful for the smoothing.

So we look now for an NN smoother whose input is a window of observations around y_i and minimize (2). The NN is now a function of the observations while the second term of $C(\lambda)$ needs to compute the second derivative with respect to the time. Direct computation of the partial derivatives with respect to the weights is not easy, because it requires the approximation of $\frac{\partial S}{\partial t}$ and $\frac{\partial^2 S}{\partial t^2}$. We propose to use an increased NN architecture in order to minimize (2). We describe now the NN smoother and then the increased architecture.

The neural network smoother uses a basic MLP architecture (R) with an input layer supporting the temporal window, l layers of non-linear hidden neurons and one linear output neuron. Starting from this basic architecture (R), a family of multi-layer perceptrons (R^λ) is generated. For each λ, the synaptic weights vector are chosen using the learning procedure described below.

The input layer uses a temporal window on the time series, which is the vector of observations ($y_{i-d'}$,....., y_{i+d}), $d' \geq |d|$. The two time lags d and d' can be equal or different. They can be both positive, in which case, d' represents the amount of information in the past used by the temporal window, d stands for the future and y_i is within the window. However, d may be negative: for example, if d = -1, y_i is not within the window, smoothing at time i uses only past information, it can be considered as a predictor of the smoothed value at time i.

In the following, we assume a fully connected architecture, but the method also holds for any MLP using prior information in its architecture. The neural network smoother is controlled by a parameter λ and minimizes (2). We use a discrete second order approximation, so (2) becomes (3):

$$C_D(\lambda) = 1/2 \sum_{i=1}^{n} [\, y_i - g(i) \,]^2 + \lambda \sum_{i=2}^{n-1} [\, g(i+1) + g(i-1) - 2\, g(i) \,]^2 \qquad (3)$$

where g is the function computed by the network.

For each λ, an optimal solution g^λ corresponds to a particular MLP, denoted by (R^λ), whose weights minimize (3). In the following we denote by $F = \{g^\lambda\}$ the family functions generated by R^λ .

In order to minimize the function (3), we will use an increased NN architecture in which g(i+1), g(i), g(i-1) will appear as ouput neurons value. This network is a TDNN [Waibel & al 1989] (see figure 1). The input layer has d+d'+3 units (for window [$y_{i-1-d'},\ldots,y_{i+1+d}$]), l hidden layers ($l \geq 0$) and 3 output units. The first hidden layer has 3 clusters of n_1 units each. The first cluster of hidden units is connected to window {i-1-d',…, i-1+d}, the second to the window shifted by one time unit, and the third to the window shifted by one more unit ; these clusters share the same weight vector. The use of time-delay inputs convert the time series into a static mapping problem. The only constraint is that each hidden layer has three clusters of units connected only to the corresponding cluster in the previous layer. In each layer, clusters share the same weight vector.

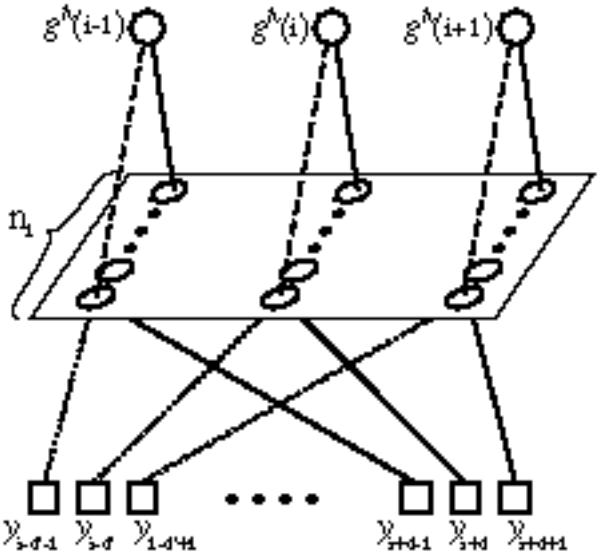

Figure 1: TDNN for smoothing.

If we denote by x_{i-1} (resp. x_i, x_{i+1}) the computed output of ouput unit 1 (resp. 2 and 3). Hence we consider the measure of roughness :

$$C_D(\lambda) = 1/2 \sum_{i=1}^{n} [\, x_i - y_i \,]^2 + \lambda \sum_{i=2}^{n-1} [\, x_{i+1} + x_{i-1} - 2\, x_i \,]^2 \qquad (4)$$

which is exactly (3) with the interpretation that x_i is g(i). The learning phase consists in minimizing the discrete approximation given in (4) by the back propagation algorithm.

When this TDNN architecture is trained, one can thus extract three sub-networks with identical architecture and weights which representsR^λ. Each sub-network has a unique

output unit. If the TDNN's input window is $[y_{i-1-d'},\ldots,y_{i+1+d}]$, then the 3 networks respectively have as input $[y_{i-1-d'},\ldots,y_{i-1+d}]$, $[y_{i-d'},\ldots,y_{i+d}]$ and $[y_{i+1-d'},\ldots,y_{i+1+d}]$, which allow them to compute respectively:

$x_{i-1} = g^{\lambda}(i-1)$, $x_i = g^{\lambda}(i)$ and $x_{i+1} = g^{\lambda}(i+1)$

Examples of smoothing obtained for various values of λ will be given in section 4. Obviously, the estimated smoothed value varies with λ.

If one now assumes that observations y_i come from an additive model of the form (1):

$$Y_i = \mu(i) + Z_i$$

The problem is then to find in the family of functions $F = \{g^{\lambda}\}$ produced by networks (R^{λ}), the smoothing which best approximates the trend $\mu()$. In the next section, we introduce a criterion which allows an optimal choice of the value of λ. This criterion will require that process Z_i satisfies mild hypothesis, which in practice usually holds for real data.

3. PARAMETER SELECTION

The smoothing g^{λ} is conditioned by the observations $(y_1,y_2,....y_n)$: in the following we denote $\hat{g}^{\lambda}(i) = \hat{g}^{\lambda}(i/y_1,y_2,....y_n)$. In order to choose the smoothing parameter λ we need some measure of discrepancy between the unknown function μ and the approximation $\hat{g}^{\lambda}$. We can consider the two following distance measurements :

• The Average Squared Error

$$ASE(\hat{g}^{\lambda}) = \frac{1}{n}\sum_{i=1}^{n}\left(\mu(i) - g^{\lambda}(i/y_1,y_2,....y_n)\right)^2$$

which is a random variable depending on the observations $(y_1,y_2,....y_n)$.

• The Mean Average Squared Error

$$MASE(\hat{g}^{\lambda}) = E\left[\frac{1}{n}\sum_{i=1}^{n}\left(\mu(i) - g^{\lambda}(i/y_1,y_2,....y_n)\right)^2\right] \qquad (5)$$

where E stands for the expectation with respect to all possible samples $(y_1,y_2,....y_n)$.
$MASE(\hat{g}^{\lambda})$ is thus independent of the particular sample and more suited for parameter selection, we used it as the mesure of discrepancy.

The MASE can be easily expressed as :

$$MASE(g^{\lambda}) = \frac{1}{n}\sum_{i=1}^{n} E\left(g^{\lambda}(i/y_1,y_2,....y_n) - \bar{g}^{\lambda}(i)\right)^2 + \frac{1}{n}\sum_{i=1}^{n}\left(\bar{g}^{\lambda}(i) - \mu(i)\right)^2 \qquad (6)$$

where $\bar{g}^{\lambda}(i) = E\left(g^{\lambda}(i/y_1,y_2,....y_n)\right)$.

The first term on the right-hand side of (6) corresponds to a variance component and the second to the square of a bias component, both of them contributing to the MASE(g^{λ}). The variance component is decreased by smoothing, but this increases the bias component. The problem is thus to choose a good compromise between these two components (which is done by minimizing (6) with respect to g^{λ} F).

Let us denote λ^* the best smoothing parameter, i.e. that value of the smoothing parameter λ such that :

$$MASE(g^{\lambda^*}) = Min_{\lambda}\ MASE(g^{\lambda})$$

The classical Leave-One-Out (LOO) cross-validation technique produces, for any real λ, an unbiased estimate CV(λ) of MASE (g^{λ}), in the case where Z_i is a white noise. In this case the samples (t_i,y_i) are independent where the t_i's are the selected observation time [Hardle, 91] [Plutowski & al, 94]. The minimum of CV(λ) is thus likely to be λ^*.

Such a cross-validation technique cannot be used in practice for neural networks, since it requires training n different networks for each value of λ (i.e. one network for each learning set $\{y_j; j = 1,...,n; j\neq i\}$, i=1…,n). Moreover we want to extend the parameter selection to accomodate an autocorrelated error sequence. Indeed, methods designed for an identically independant distributed case can fail in the setting of smoothing with dependant errors $\{Z_i\}$. For example, cross validation can select too small a value of λ, the reason being that this method interprets the existing correlation in $\{Z_i\}$ as part of the trend $\{\mu(i)\}$.

In the following we prove that an extended cross-validation criterion which is computationall easy can be designed for NN. The use of such a criterion is very helpful in the sense that only requires computing its value for each λ, and needs no relearning.

We introduce now the new criterion SC(λ) which allows determining for each λ an unbiased estimator of the MASE(g^{λ}).

Let us introduce some notations related to network (R^{λ}):

• W^{λ}_{jk} is the weight from input cell k ($-d' \leq k \leq d$) to hidden cell j ($1 \leq j \leq n_1$) in the first hidden layer (with n_1 units),

• at time i, the input of the network is the temporal window ($y_{i-d'}, \ldots, y_{i+d}$). Unit j in the first hidden layer receives the weighted input:

$$A_j(i) = \sum_{k=-d'}^{d} y_{i+k} W^{\lambda}_{jk} ,$$

• in the following we will assume that $\{Z_i\}$ satisfies assumption (H1):

$\{Z_i\}$ is a stationary zero mean autocorrelated process, with zero auto-covariance coefficients after a known order M. (H1)

M must be chosen, depending upon the noise model. Note that, when $\{Z_i\}$ is a white noise, M = 0.

Suppose that the network is presented, at its input layer, with a truncated temporal window where all observations $\{ y_{i+k} ; -M \leq k \leq M \}$ are 0, all other observations being unchanged, the weighted input to hidden unit j then is:

$$A^{*}_j(i) = \sum_{\substack{k=-d' \\ |k|>M}}^{d} y_{i+k} W^{\lambda}_{jk} \qquad j = 1, \ldots, n_1$$

• let us denote $B_j(i)$ the information loss for cell j:

$$B_j(i) = A_j(i) - A^{*}_j(i) = \sum_{\substack{k=-d' \\ |k| \leq M}}^{d} y_{i+k} W^{\lambda}_{jk}$$

• the computed output of network R^{λ} can be expressed by a function of the form: $F^{\lambda}(u_1, \ldots, u_j, \ldots, u_{n_1})$ where u_j represents the output of the hidden unit j of the first layer : $u_j = f (A_j(i))$

where f() is the transfer function of the hidden units.

Under these notations, the outputs of R^λ, when presented with the complete (resp. truncated) temporal window, are $g^\lambda(i)$ (resp. $\overset{\lambda}{g}_M(i)$) :

$$g^\lambda(i) = F^\lambda(f(A_1(i)) , ... , f(A_{n_1}(i)))$$
$$\overset{\lambda}{g}_M(i) = F^\lambda(f(\overset{*}{A}_1(i)), ... , f(\overset{*}{A}_{n_1}(i))).$$

Under these notations we define the new criterion SC(λ) as:

$$SC(\lambda) = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \overset{\lambda}{g}_M(i)\right)^2 - \frac{1}{n}\sum_{i=1}^{n}\left(\sum_{j=1}^{n_1} B_j(i)Q_j(i)\right)^2 \qquad (7)$$

where $Q_j(i)$ is the partial derivative of $F^\lambda(f(u_1)$,···, $f(u_j)$,···, $f(u_{n1}))$ with respect to u_j computed at vector $(f(\overset{*}{A}_1(i)) , ..., f(\overset{*}{A}_j(i)),... , f(\overset{*}{A}_{n_1}(i)))$:

$$Q_j(i) = \frac{\partial F^\lambda}{\partial u_j}(f(\overset{*}{A}_1(i)), ... , f(\overset{*}{A}_{n_1}(i))) .$$

SC(λ) has two terms: the first one deletes observations in the inner window. The second one is a corrective term to take into account the auto-correlations of $\{Z_i\}$.

The procedure to compute SC(λ) for every λ, is as follows:

PROCEDURE TO COMPUTE SC(λ)
- train the TDNN with cost function (4), and learning set $\{y_i; i = 1,...,n\}$;
- present the complete series to the network and evaluate $\{A_j(i), j = 1,\ldots,n_1\}$;
- present the truncated series to the network and evaluate:
  - in the forward pass: $\{\overset{*}{A}_j(i)\}$ and $\{B_j(i)\}$, $j = 1,\ldots,n_1$,
  - in the backward pass: $\{Q_j(i)\}$, $j = 1,\ldots,n_1$;
- apply formula (7).

SC(λ) is thus easy to compute and requires training the network only once.

Formula (7) simplifies when R^λ has only one hidden layer. Let us denote $\overset{\lambda}{W}_j$ the weight from hidden cell j to the output cell. In this case:

$$SC(\lambda) = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - g_{M(i)}^{\lambda}\right)^2 - \frac{1}{n}\sum_{i=1}^{n}\left(\sum_{j=1}^{n1} f'(A_j^*)\, B_j(i)\, W_j^{\lambda}\right)^2 \qquad (8)$$

The following result shows that under mild hypothesis up to a constant, $SC(\lambda)$ is an unbiased estimator of $MASE(g^{\lambda})$.

Under the following assumptions

- $\{Z_i\}$ is a stationary autocorrelated process with zero mean and zero auto-covariance coefficients after a known order M. (H1)
- $g_{M(i)}^{\lambda}$ is independent of the observation y_i. (H2)

then there exists a neighbourhood V of λ^* such that

for all λ V then $E(SC(\lambda))$ $\gamma_0 + MASE(g^{\lambda})$ (9)

where γ_0 is the variance of Z_i

This result allows us to select the best smoothing parameter λ. If M is known, the minimum of $SC(\lambda)$ provides a good smoothing under the assumptions on $\{Z_i\}$. However, if M is unknown, one can compute $SC(\lambda)$ for various values of M and select the best one according to any given criterion [1].

The proposed result relies on hypothesis (H2). We discuss in the annex how the learning procedure of (R^{λ}) can be organized in order to raise the validity of hypothesis (H2).

PROCEDURE TO COMPUTE λ^*

- compute $SC(\lambda)$ for all possible values of λ,
- approximating $E[SC(\lambda)]$ with $SC(\lambda)$, we thus have from (9) :

  $SC(\lambda)$ $\gamma_0 + MASE(g^{\lambda})$
- determine the minimum λ^{**} of $SC(\lambda)$
- from (9), we then have:

  λ^* λ^{**}

Determination of λ^* thus requires computing the values of $SC(\lambda)$ for all possible values of λ, which requires training one network for each λ.

[1] For example, one can look at the spectra of g_{λ^*}, for each M; and pick that M which best eliminates noise, i.e. high frequencies.

The proposed procedure involves testing the trained network R^{λ} using the learning set. It does not require the use of an independant validation set. The procedure uses both input data (temporal windows of observations) and truncated input data (temporal window of observations from which some observations are omitted).

As we saw in section 3, the Neural Network can be used as a predictor, by taking d=-1. The input to R^{λ} is then window $(y_{i-d'}, \ldots, y_{i-1})$. Value $g^{\lambda}(i)$, computed by R^{λ} is then a prediction at time i. Since y_{n+1} is not known at time t_n, then the $SC(\lambda)$ criterion is an unbiased estimator of the MASE computed at the selected time $(t_1, \ldots, t_{n-1})$. In this case, the preceding procedure does not, *a priori*, guarantee an unbiased estimation for MASE computed up to selected time t_n [Plutowski & al 94]. But for large n we can suppose that the learning procedure trained with: $(y_1, \ldots, y_{n-1})$ will compute a function $g^{\lambda}()$ which approximates the function trained with the learning set $(y_1, \ldots, y_n)$. In this case, we can thus select the best possible prediction at time t_n.

4. NN SMOOTHING ANALYSIS

Kernel functions are functions whose integral is equal to 1, their principal characteristics are that they become negligible outside a finite range. In practice, they are defined on a window [-m, M]. A given kernel K allows the generation of a family of kernel functions using a smoothing parameter h: $K^h(t) = \frac{1}{h} K\left(\frac{t}{h}\right)$. Each kernel function K^h is thus defined on a window [-hm, hM]. For a given h, hence for a given kernel function K^h, the estimated kernel smoothing $\hat{\mu}^h(i)$ is defined, at each time i, as a moving average of the observations $\{y_j; j = 1,\ldots,n\}$:

$$\hat{\mu}^h(i) = \sum_{j=1}^{n} y_j W_{ji} \quad \text{where } W_{ij} = \frac{K^h(i-j)}{\sum_{k=1}^{n} K^h(i-k)}$$

$\hat{\mu}^h$ is a linear smoother whose coefficients W_{ij}^h depend on the time interval (i-j) and not on the observations $\{y_j; j=1,\ldots,n\}$. In practice, coefficients W_{ij}^h are almost zero when i-j >hM or j-i > hm. Hence, h controls the width of the moving average window:

Another smoothing method is to choose from the family of twice continuously differentiable functions, such that $\int_{-\infty}^{+\infty} S''(t)^2 dt$ exists, the function S which minimises the cost function (2).

The weighting parameter λ controls the amount of smoothing, so that increasing λ leads to a smoother estimate. Theoretical results are available on such minimisation and show that, for a given λ, the function S^{λ} minimising $C(\lambda)$ is a cubic spline [Reinsch 67; Eubank 88]. As a function of t spline smoothing applies in the time domain and S^{λ} is a twice continuously differentiable function of t.

A relation between kernel and spline smoothing can be found, when we consider the symmetric spline kernel $K(t) = 0.5 \exp\left(-\frac{|t|}{\sqrt{2}}\right) \sin\left(0.25\pi + \frac{|t|}{\sqrt{2}}\right)$. Silverman [Silverman 84] has shown that, for time series, assuming that the number of observations n is large and λ is small enough, then:

$$S^{\lambda}(i) \quad \hat{\mu}^{h}(i) = \sum_{i=1}^{n} y_j \frac{K^{h}(i-j)}{\sum_{k=1}^{n} K^{h}(i-k)} \text{ with } h = \lambda^{0.25}$$

where S^{λ} is the function S which minimises $C(\lambda)$. Then, the smoothing parameter λ of S^{λ} and the window size of the kernel h are linked by the relationship $h=\lambda^{0.25}$. This clearly shows that kernel smoothing operates in the time domain, since it is the optimal spline function S^{λ} over the discrete time unit i.

Kernel smoothing is related to NN-smoothing since they both use a temporal window of observations. It is also a particular NN smoother with a single layer of weights. But NN smoothers are explicit functions of the observations since the weights are computed through learning, using all the observations $\{y_j ; j=1...n\}$, and depend on them. NN performs a smoothing in the state space using a cost function $C_D(\lambda)$ (4) which acts in the time domain. So the NN smoother combines two representations : the state space domain and the time domain. The NN network directly minimizes (4) which is the discrete form of (2) and gives the optimal smoothing directly for each discrete time unit i and not on the continuous time domain as the spline smoother does.

The advantages of the NN smoothers derive from their non linearity, the use of two different domains of representation and the direct minimization of a specific cost function dedicated to discrete time series.

5 . THE EXPERIMENTS

The goal of this section is to illustrate the theoretical results presented in the previous sections with some appropriate examples. The main results are: first that the Smoothing

Criterion (SC) gives the optimal smoothing under some hypotheses concerning $\{Z_i\}$, second that the smoothed function can be learnt in order to predict new values of the trend. The experiments deal with simulated data. As an example, we have simulated a time series of equally spaced data satisfying model (1) (figure 2):

$$y_i = \mu(i) + Z_i$$
$$\mu(i) = \left(\sin\left(2\pi t_i^3\right)\right)^3$$
$$\{Z_i = \varepsilon_i + 0.25\, Z_{i-1}\} \qquad (10)$$

with: $\varepsilon_i \approx N(0\,;\,0{,}1)$, i=1..300 and the time of the observation y_i is equal to $i/300$.

The stochastic process Z_{t_i} is a first-order autoregressive sequence:

$$\gamma_k = \mathrm{cov}(Z_i, Z_{i+k}) = 0.25^k \gamma_0 \qquad (11)$$

Clearly $\gamma_k \quad 0$ for k>2.

All the observations of time series $\{y_1, y_2, \ldots, y_{300}\}$ are correlated (Fig.2). Nevertheless it can be assumed that the auto-covariance coefficients of the stationary process are equal to zero after a lag of 1 or 2 (M=1 or M=2). All the MLP have the same architecture which has been determined empirically. This architecture has to be a good compromise between the number of parameters of the MLP and the order of M. This means that d+d'+1 and M must be such that there is enough information to correctly compute SC. The smoothing is controlled by the smoothing parameter λ, and the window size (d+d'+1) only gives a temporal context. After a few trials, we use a fully connected MLP with 21 input units and 1 hidden layer (l=1) with 3 hidden units.

We tested our method with three different temporal windows :

- Symmetric window d=d'=10
- Non symmetric window d=5, d'=15
- Predictor window d'= 21, d=-1 (this network predicts the next smoothed value).

In each case we apply our NN smoother in order to generate a set of smoothing functions: $\{g^\lambda / \lambda=1...30\}$. We use SC($\lambda$) to choose the best smoothing under three different hypotheses M=1, M=2 and M=3. In order to prove the efficiency of this technique we have also computed the true error : the Average Squared Error (ASE) which is of the form :

$$ASE(g^{\lambda}) = \frac{1}{n}\sum_{i=1}^{n}(\mu_i - g^{\lambda}(i))^2 \qquad (12)$$

We denote by λ the parameter proposed by the SC function such that :

$$\lambda = \arg \min_{\lambda=1 \quad 30} \{SC(\lambda)\}$$

Table 1 compares, for each temporal window and for the different hypotheses on M, the error $ASE(g^{\lambda^*})$ to the smallest error $\min_{\lambda=1 \quad 30} \{ASE(\lambda)\}$ which represents the best smoothing available from the family of smoothers generated by the neural network architecture. Clearly the values of $ASE(g^{\lambda^*})$ are close to the minimum $\min_{\lambda=1 \quad 30} \{ASE(\lambda)\}$ and the SC gives a good estimate of the ASE. Table 1 shows that the best performances are obtained for M=1 or M=2 which fits the theoretical value of γ_K 0 for k>2.

	M	λ	$ASE(g^{\lambda})$	MIN_{λ} $ASE(g^{\lambda})$
d = d' =10 (symmetric)	0 1 2	2 10 12	0.0195 0.0168 0.0161	0.0114
d = 5 , d' = 15 (non symmetric)	0 1 2	2 10 14	0.0176 0.0143 0.0130	0.0112
d = -1 , d' = 20 (predictor)	0 1 2	0 3 6	0.0205 0.0194 0.0216	0.0186

Table 1 : performances provided by the SC for the symmetric, non symmetric and predictor windows. λ is the value which minimizes SC(λ) assuming as hypothesis : M=0, M=1 or M=2.

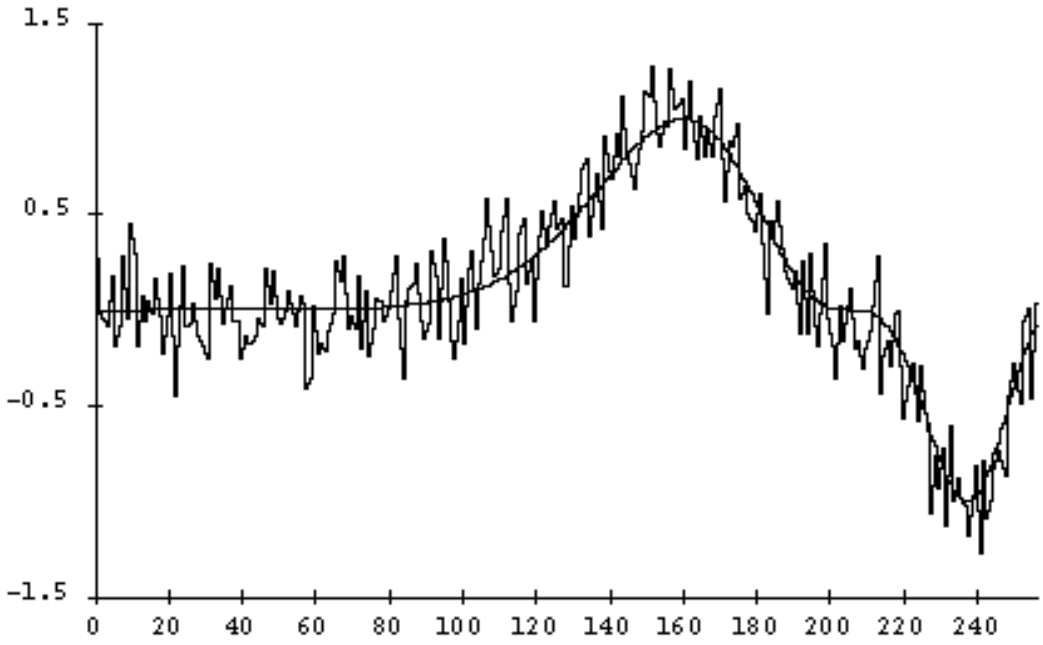

Figure 2: data set $y_i = (\sin(2\pi t_i^3))^3 + Z_i$; $Z_i = \varepsilon_i + 0.25\,\varepsilon_{i-1}$; $\varepsilon_i \approx N(0;0.1)$, $\mu_i = (\sin(2\pi t_i^3))^3$ and $t_i = 1/300$. The horizontal axis represents time i, and the vertical axis represents signal amplitude. The smooth curve represents the simulated curve μ_i and the other represents the noisy curve y_i.

Figure 3 gives two different smoothings obtained for two different values of λ, using the asymmetric window. Figure 4 presents SC(λ) for the asymmetric window assuming that M=1, in this case the minimum is reached for $\lambda = 10$.

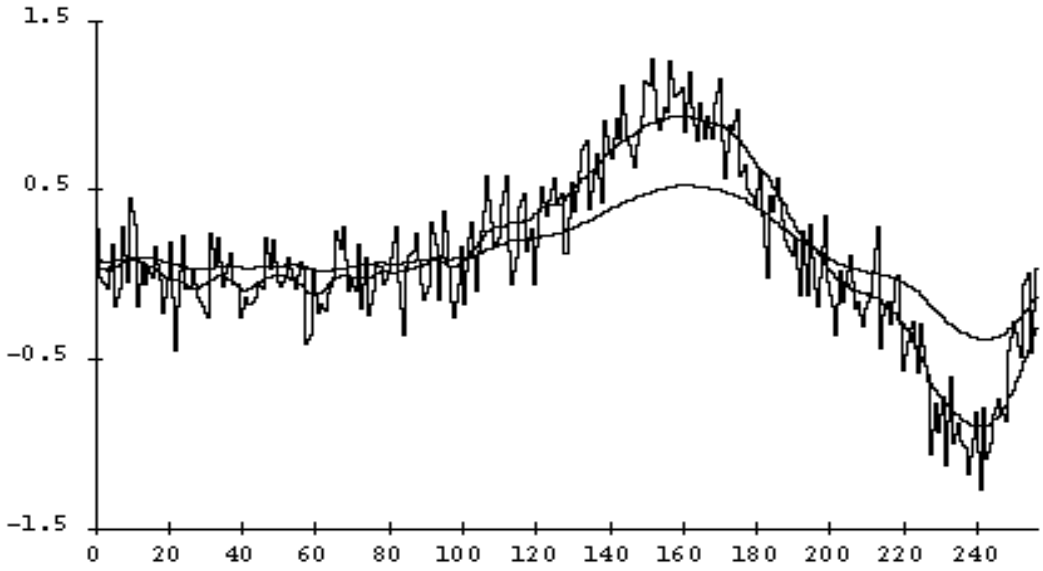

Figure 3: data set $y_i = (\sin(2\pi t_i^3))^3 + Z_i$; $Z_i = \varepsilon_i + 0.25\,\varepsilon_{i-1}$; $\varepsilon_i \approx N(0;0.1)$. The horizontal axis represents time i, and the vertical axis represents signal amplitude. The noisy curve represents y_i and the two other curves are smoothing obtained in the asymmetric case (d=5, d'=15) with M=1. The over-smoothed curve corresponds to the estimated curve $g^{\lambda}(i)$ for $\lambda = 30$. The second smoothed curve corresponds to the optimal smoothing $g^{\lambda^*}(i)$, $\lambda^* = 10$ has been selected by the SC(λ) criteria.

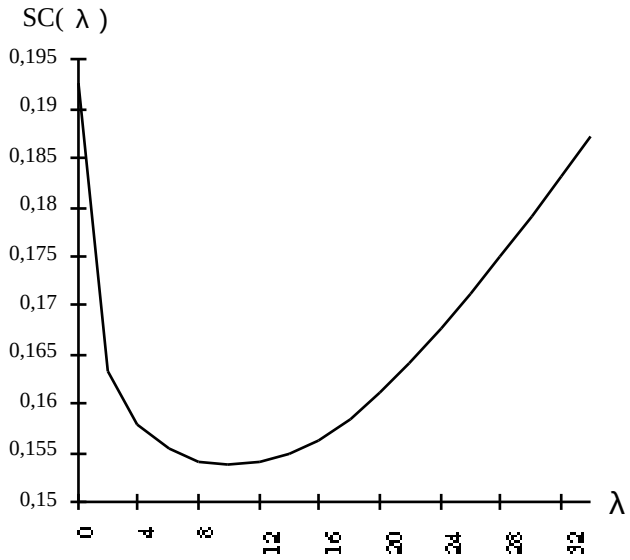

Figure 4 : graphical representation of SC(λ) in the asymmetric case (d=5, d'=15) under the hypothesis M=1. The minimum is reached for $\lambda^* = 10$, the smoothed curve corresponding to this value is given in figure 3.

When dealing with observations where γ_k cannot be computed it is necessary to choose an optimal value for M. This can be done by applying the SC criterion for different values of M, and then looking at the resulting smoothings and their power spectrums (Fourier transforms). Figure 5 gives the power spectrum of the smoothed curves g^λ for M=0, M=1, and M=2 in the asymmetric case. Evidently the use of M=1 or M=2 removes the noise of the data. In the light of these experiments one can choose M=1 as a possible hypothesis for the order of the stochastic process and accept the decomposition :

$$Y_i = g^\lambda(i) + \tilde{Z}_i$$

with $\lambda = 12$ and $\tilde{Z}_i = (y_i - g^\lambda(i))$

When using the prediction window the same results can be observed for M=1 and $\lambda = 3$ $g^\lambda(i)$ acts as a predictor of the trend. $\tilde{Z}_i$, which is quasi stationary, can be modeled using a dedicated MLP.

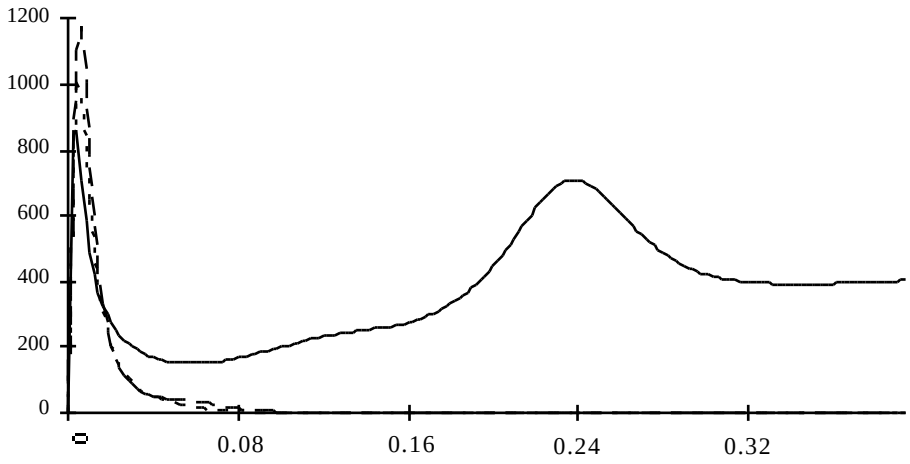

Figure 5 : Power spectrum of three curves g^{λ} in the asymmetric case (d=5, d'=15). Each λ^* has been selected using the SC criteria under three different hypotheses M=0, M=1, M=2. The horizontal axis represents the frequencies and the vertical axis represents energy amplitudes. The solid line represents the curve obtained for M=0. It is clear that the peak of energy at frequency 0.24 indicates that the noise added to the simulated data has not been completely removed. This shows that the hypothesis M=0 corresponding to a white noise hypothesis is not valid. The two dashed lines correspond to the power spectrum obtained under the hypothesis M=1 and M=2, they are similar and evidently remove the noise.

6.CONCLUSION :

This paper discusses the use of Neural Networks techniques for non-parametric estimation of the "smooth" function $\mu(i)$ of a time series of the form $Y_i = \mu(i) + Z_i$ where $\mu(i)$ is a non-random function and Z_i a stochastic process. The study is general and only assumes that the auto-covariance coefficients of the stationary process Z_i are null after a known order M (γ_k= cov (Z(t), Z(t+k)) 0 for k > M). We propose a neural architecture which allows the estimation of the trend $\mu(i)$ from the data. The learning procedure is related to spline smoothing and kernel regression but the whole process can be considered as a non-linear smoother. We then introduce a general smoothing criterion SC, which allows the selection of the optimal smoothing λ^*.

Based on experimental results, the Neural Network Smoother appears to be a flexible smoothing technique and the SC an efficient criterion for model selection. Moreover, when compared with classical methods (kernels, splines) NN smoothing provides an easy way to forecast the trend of a time series (predictor window).

NN smoothing can provide a neural methodology similar to Box and Jenkins. Box and Jenkins' forecasting approach first removes the trend. This is done by successively differenciating the time series until its time plot appears to be stationary and then proceeding to the model identification. This identification is made by fitting the

parameter of an ARMA process. We have seen that the same processing can be made using neural networks: first the trend is removed by using a forecast window ($d < 0$). After the removal of the predicted trend, the remaining time series of residue: $(y_i - g^{\lambda}(i))$ is quasi stationary and can be easily modeled by static or recurrent NN. A second neural network can be used in order to estimate Z_i. Training both architectures, the first one predicting the trend and the other predicting the residue, allows a neural decomposition of the initial series (y_i, $i=1..n$) in the form of (1). It thus follows that the modular architecture can be considered as a predictor of the initial series. Some preliminary comparisons with real data have been made and show that a first estimation of the trend is a valid approach [Kouam 1993].

7. REFERENCES

Bishop, C. M. (1990). Curvature-driven smoothing: a learning algorithme for feed-forward networks. IEEE Transactions on Neural Networks, 4 No. 5 1993, 882-884.

Diggle, P. J., and Hutchinson, M.F. (1989). On spline smoothing with autocorrelated errors. Austral J. Statist, 31(1) 1989, 166-182 .

Eubank, R.L. (1988). Spline smoothing and nonparametric regression. Marcel Dekker, New Yok

Härdle, W., (1991). Smoothing techniques. Springer series in statistics. Springer Verlag.

Kouam, A. (1993). Approches connexionnistes pour la prevision des series temporelles. Thèses Université de Paris Sud Centre D'orsay 1993.

MacKay, D. (1992). Bayesian interpolation. Neural computing 4(3): 415-447.

Moody, J.E. (1992). The *effective* number of parameter: an analysis of generalization and regularization in nonlinear systems. In Moody, J.E., Hansen, S.J., and Lipmann, R.P., (eds), Advanced in neural information processing systems 4, San Mateo, CA: Morgan Kaufmann Publishers.

Plutowski M., S. Sakada, and H. White (1994). Cross validation estimates IMSE. In Cownd J.D., Tesarau G., Alspector J., (eds), Advanced in Neural Information Processing Systems 6, Morgan Kaufman Publishers.

Silverman, B.W.(1984). Spline smoothing: the equivalent variable kernel method. Annals of statistics, 12, 898-916.

Wahba, G. (1990). Spline models for observational data. SIAM.

Waibel,A., T. Hanazawa, G. Hinton, K. Shikano, and K. Lang. (1989). Phoneme recognition using Time-Delay Neural Networks. IEEE Trans. Acoust., Speech, & Signal Proc. 37(3): 328-339.

Weigend, A.S. and N.A. Gershenfeld, eds (1993). Time series prediction: forecasting the future and understanding the past. Santa Fe Institute Studies in the sciences of compexity, Proc. vol. XV. reading, MA: Addisson-Wesley.

Annex

We use in the following the same notations as in section 3. Thus g^{λ} is the function implemented by network R^{λ}, λ^* is the value such that:

$MASE(g^{\lambda}) = Min_{\lambda} MASE(g^{\lambda})$.

We will show that the relation (9) is valid under the H1 and H2 assumptions:
H1: $\{Z_i\}$ is a stationary autocorrelated process, with zero mean and auto-covariance coefficients after a known order M.
H2 : $\overset{\lambda}{g}_M(i)$ does not depend on the random variable y_i.
H1 is often a valid hypothesis, M representing the memory of the $\{Z_i\}$process. As for the H2 hypothesis: the expression of $\overset{\lambda}{g}_M(i)$ depends directly on $\{y_{i+k}, |k| \geq M\}$ variables which are independant of y_i.because of H1. The statistical dependancy of y_i and $\overset{\lambda}{g}_M(i)$ relies on the learning procedure. There are various ways to enforce the hypothesis H2, through learning. One can add a regularization term to cost function (4) (e.g. a weight decay), in order to constrain the solution space so that the solution $\overset{\lambda}{g}_M$ will be robust to slight changes in the observations, for example changes in the window $\{y_{i+k}, |k| \leq M\}$ if M is small enough. Furthermore this hypothesis will be reinforced if the size of the time series is large enough.
Let us define $G(\lambda)$, a random function depending on the observations $(y_1,y_2,....y_n)$:

$$G(\lambda) = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \overset{\lambda}{g}_M(i)\right)^2 \quad (13)$$

The expectation of $G(\lambda)$ with respect to all possible samples $(y_1,y_2,....y_n)$ is then :
$E(G(\lambda)) = E$**Erreur!**$^2)$
= γ_0 + **Erreur!**))+ **Erreur!**)
= γ_0 + **Erreur!**)+ **Erreur!**) (14)
where γ_0 is the variance of Z_i.
Under assumption (H2) : $Z_i = y_i - \mu(i)$ and $\overset{\lambda}{g}_M(i) - \mu(i)$ are independent variables.
Hence:
E **Erreur!**) = 0.
We thus have:
$E(G(\lambda)) = \gamma_0$ + **Erreur!**). (15)
Using notations of section 3 and the Taylor expansion formula we have:
$g^{\lambda}(i) - \overset{\lambda}{g}_M(i) = F^{\lambda}(f(A_1(i)) , ... , f(A_{n_1}(i))) - F^{\lambda}(f(A^*_1(i)), ... , f(A^*_{n_1}(i)))$

$$= \sum_{j=1}^{n1} B_j(i)Q_j(i) + o(\|B(i)\|) \quad (16)$$

where j stands for a unit of the first hidden layer, $Q_j(i) = \frac{\partial F^{\lambda}}{\partial u_j}(f(A^{*}_{1}(i)), \ldots , f(A^{*}_{n1}(i)))$

$B(i) = (B_1(i), .,B_j(i),.. , B_{n_1}(i))$.

Since: $g^{\lambda}_{M}(i) = g^{\lambda}(i) - \sum_{j=1}^{n1} B_j(i)Q_j(i) - o(\|B(i)\|)$

then (16) gives:

$$E(G(\lambda)) = \gamma_0 + \textbf{Erreur!})$$

$$+ \frac{2}{n} \sum_{i=1}^{n} E\left[(\mu(i) - g^{\lambda}(i)) \left(\sum_{j=1}^{n1} B_j(i)Q_j(i) + o(\|B(i)\|) \right) \right]$$

$$+ \frac{1}{n} \sum_{i=1}^{n} E\left[\left(\sum_{j=1}^{n1} B_j(i)Q_j(i) \right)^2 \right]$$

$$+ \frac{2}{n} \sum_{i=1}^{n} E\left[o(\|B(i)\|) \sum_{j=1}^{n1} B_j(i)Q_j(i) \right] +$$

$$\frac{1}{n} \sum_{i=1}^{n} E\left[o(\|B(i)\|)^2 \right]$$

The last term is of order 2, thus negligible.

Term $\frac{1}{n} \sum_{i=1}^{n} E\left[\left(\sum_{j=1}^{n1} B_j(i)Q_j(i) \right)^2 \right]$ does not depend on the unknown function $\mu(i)$

and is easily computed; so we define the SC criterion by:

$$SC(\lambda) = G(\lambda) - \frac{1}{n}\sum_{i=1}^{n}\left(\sum_{j=1}^{n1} B_j(i)Q_j(i)\right)^2$$

The second term of $SC(\lambda)$ is a correcting term of $G(\lambda)$. We have :

$$E(SC(\lambda)) = \gamma_0 + MASE(g^\lambda)$$
$$+ \frac{2}{n}\sum_{i=1}^{n} E\left[(\mu(i) - g^\lambda(i))\left(\sum_{j=1}^{n1} B_j(i)Q_j(i) + o(||B(i)||)\right)\right]$$
$$+ \frac{2}{n}\sum_{i=1}^{n} E\left[o(||B(i)||)\sum_{j=1}^{n1} B_j(i)Q_j(i)\right] \qquad (17)$$

The last term of the right hand side is negligible with respect to the correcting term. Let us denote by BIAS the third term of the right hand side of (17) :

$$BIAS = \frac{2}{n}\sum_{i=1}^{n} E\left[(\mu(i) - g^\lambda(i))\left(\sum_{j=1}^{n1} B_j(i)Q_j(i) + o(||B(i)||)\right)\right]$$

It is difficult to evaluate the order of magnitude of the BIAS term, but taking some precautions during the learning phase can lead it to be small :

1- By using a weight-decay technique in the learning phase to verify the asumption (H2), we keep the weights homogeneous and small, consequently $\sum_{j=1}^{n1}(B_j(i)Q_j(i))+o(||B(i)||)$ is a small value. The parameter weighting the regularisation term is the same for each (R^λ). This parameter is chosen small enough, so as not to compete with the smoothing term of $C_D(\lambda)$.

2 - We suppose that the size of the time series is large enough and that the learning of each (R^λ) is achieved using the same conditions : weights initialisation, learning rate.

If precautions (1) and (2) are used, we can asume that g^λ is continuous with respect to λ. Using the continuity of g^λ, when λ varies from 0 to infinity, g^λ will be continuously

distorted from over smoothing to under smoothing. As a consequence we will assume that :

there exists a neighbourhood V of the optimal value λ^* and a fixed time lag T around the observations such that for each λ in V we have $\sum_{j=i-T}^{i+T} \left[\mu(i) - g^{\lambda}(i) \right] = o(||B(i)||)$

During this time lag the mean value of $\left(\sum_{j=1}^{n1} B_j(i) Q_j(i) + o(||B(i)||) \right)$ is near constant.

So from these two considerations the BIAS term can be considered of the order :

$$\frac{2}{n} \sum_{i=1}^{n} E\left(o(||B(i)||) \sum_{j=1}^{n1} B_j(i) Q_j(i) \right)$$

and can be neglected with respect to the correcting term of SC(λ). The simulations presented in section 5 confirm the present assumptions : The magnitude of the BIAS term is 10% of the correcting term.

From relation (17) and neglecting the two last terms we have :

$$E(SC(\lambda)) \quad \gamma_0 + MASE(g^{\lambda})$$