Automatic selection of tuning parameters in wind power prediction

Lasse Engbo Christiansen (`lec@imm.dtu.dk`)
Henrik Aalborg Nielsen (`han@imm.dtu.dk`)
Torben Skov Nielsen (`tsn@imm.dtu.dk`)
Henrik Madsen (`hm@imm.dtu.dk`)
Informatics and Mathematical Modelling
Technical University of Denmark
DK-2800 Kongens Lyngby

May 22, 2007

Report number:	IMM-Technical Report-2007-12
Project title:	Intelligent wind power prediction systems
PSO Project number:	FU 4101
Ens. journal number:	79029-0001

Contents

Summary

This document presents frameworks for on-line tuning of adaptive estimation procedures. First, introducing unbounded optimization of variable forgetting factor recursive least squares (RLS) using steepest descent and Gauss-Newton methods. Second, adaptive optimization of the bandwidth in conditional parametric ARX-models.

It was found that the steepest descent approach was more suitable in the examples considered. Further a large increase in the stability when using the proposed transformation of the forgetting factor as compared to the standard approach using a clipper function is observed. This becomes increasingly important when the optimal forgetting factor approaches unity.

Adaptive estimation in conditional parametric models are also considered. A similar approach is used to develop a procedure for on-line tuning of the bandwidth independently for each fitting point. Both Gaussian and tri-cube weight functions have been used and for many applications the tri-cube weight function with a lower bound on the bandwidth is preferred.

Overall this work documents that automatic tuning of adaptiveness of tuning parameters is indeed feasible and makes it easier to initialize these classes of systems, e.g. when predicting the power production from new wind farms.

1 Introduction

The wind power forecasting system developed at DTU - the Wind Power Prediction Tool (WPPT) - predicts the power production in an area using a two stage approach. First meteorological forecasts of wind speed and wind direction are transformed into predictions of power production for the area using a power curve like model. Then the final power prediction for the area is calculated using an optimal weight between the currently observed production in the area snd the production predicted using the power curve model. Furthermore, some adjustments for diurnal variations are carried out (See Madsen et al. (2005) for details).

The power curve model is a conditional parametric model whereas the weighting between observed and predicted production from the power curve is modelled by a traditional linear model. For on-line applications it is advantageous to allow the model estimates to be modified as data becomes available, hence recursive methods are used to estimate the parameters/functions in WPPT.

No model estimation is required prior to installing WPPT at a new location, however a number of tuning parameters have to be selected. These include the forgetting factor of the recursive estimation and the bandwidth used in the conditional parametric representation of the power curve.

This report contains the derivations of two algorithms for automatic selection of the tuning parameters. The description of both algorithms are followed by examples with simulated data representing reoccuring problems in wind power prediction. The first part is on to tuning of the forgetting factor and the second part is on tuning of the bandwidth.

Each part has its own discussions and the report includes a combined conclusion in the end.

2 Unbounded optimization of variable forgetting factor RLS

2.1 Introduction

Recursive least squares (Ljung and Söderström, 1983) are successfully applied in many applications. Often exponential forgetting with a fixed forgetting factor is used but in some cases there is not enough information to chose the optimal forgetting factor and it may vary with time due to changes in the model. In such cases it might be appropriate to use an extended RLS algorithm incorporating a variable forgetting factor (VFF). Among the first to suggest a variable forgetting was Fortescue et al. (1981). They suggested a feed-back from the squared prediction error to the forgetting factor such that a large error results in a faster discounting of the influence of older data. The basic drawback with exponential forgetting is its homogeneity in time. One effect of that is covariance blowup, when certain parts of the covariance matrix grows exponentially due to lack of new information about the corresponding parameters (Fortescue et al., 1981). An alternative to exponential forgetting is linear forgetting where variation of the parameters are described by a stochastic state space model (Peters and Antoniou, 1995). A survey of general estimation techniques for time-varying systems is found in Ljung and Gunnarsson (1990).

Numerous extended RLS algorithms with variable forgetting factors are available in the literature, these includes gradient of parameter estimates (Cooper, 2000), steepest descent algorithms (Malik, 2003; So et al., 2003), and a Gauss Newton update algorithm (Song et al., 2000). See also (Haykin, 1996).

To be a valid RLS algorithm the forgetting factor has to fulfill: $0 < \lambda \leq 1$. The gradient methods, steepest descent and Gauss Newton, uses sharp boundaries for the forgetting factor. This may cause problems as the estimate of the gradient (and Hessian) do not incorporate this. In this section a new unbounded formulation of the steepest descent update of the forgetting factor is presented and simulations are used to show that this approach is more stable. The extension to a Gauss Newton update of the forgetting factor is presented and discussed.

2.2 Revised SD-RLS

2.2.1 Unbounded optimization of the forgetting factor

As mentioned above the implemented upper bound for the forgetting factor is often made using a sharp boundary also called a clipper function. This is problematic since the underlying algorithms essentially are developed for unbounded optimization problems. The clipper function has been observed to destabilize the optimization of the forgetting factor causing unwanted rapid reductions of the forgetting factor after hitting the boundary at unity. To circumvent this we propose a new formulation optimizing a transformed forgetting factor, g_t, in $\lambda(g_t)$ instead of optimizing the forgetting factor, λ_t. The function $\lambda(g)$ must be an everywhere increasing function preferably mapping the real axis to the interval: $[\lambda_-; \lambda_+]$.

Inspired by the relation between the effective number of observations N_{eff} (Also called *memory time constant* (Ljung, 1999)) and the forgetting factor:

$$\lambda = 1 - \frac{1}{N_{\mathit{eff}}} \quad , \; N_{\mathit{eff}} > 1 \tag{1}$$

we propose the sigmoid:

$$\lambda(g) = 1 - \frac{1}{N_{min} + exp(g)} \quad , \; g \in \mathbb{R} \text{ and } N_{min} > 1 \tag{2}$$

where N_{min} is giving the lower bound on λ and the upper bound is unity. The exponential is incorporated to allow $g_t \in \mathbb{R}$.

2.2.2 Deriving the general algorithm

The standard RLS algorithm (Ljung and Söderström, 1983) with input $\boldsymbol{x}_t$, observations, y_t, using the inverse correlation matrix $\boldsymbol{P}_{t-1}$, and using $\lambda(g_{t-1})$ is given by:

$$\begin{aligned}
\boldsymbol{k}_t &= \frac{\boldsymbol{P}_{t-1}\boldsymbol{x}_t}{\lambda(g_{t-1}) + \boldsymbol{x}_t^T \boldsymbol{P}_{t-1}\boldsymbol{x}_t} && (3)\\
\xi_t &= y_t - \boldsymbol{x}_t^T \boldsymbol{\theta}_{t-1} && (4)\\
\boldsymbol{\theta}_t &= \boldsymbol{\theta}_{t-1} + \boldsymbol{k}_t \xi_t && (5)\\
\boldsymbol{P}_t &= \lambda^{-1}(g_{t-1})(\boldsymbol{I} - \boldsymbol{k}_t \boldsymbol{x}_t^T)\boldsymbol{P}_{t-1} && (6)
\end{aligned}$$

where $\boldsymbol{k}_t$ is the gain, ξ_t is the á priori prediction error, and $\boldsymbol{\theta}_t$ is the vector of parameter estimates.

To adjust the forgetting factor the ensemble averaged cost function (Haykin, 1996, Sec. 16.10)

$$J_t = \frac{1}{2} E[\xi_t(\boldsymbol{\theta})^2] \tag{7}$$

is used. The first order derivative with respect to g is needed in order to derive a steepest descent algorithm:

$$\nabla_{g,t} = \frac{\partial J_t}{\partial g} = E\left[\frac{\partial \xi_t(\boldsymbol{\theta})}{\partial g} \xi_t(\boldsymbol{\theta})\right] \tag{8}$$

Note that here we take the derivative with respect to g without a time index. This corresponds to considering the situation where g, and thereby the forgetting factor, is changing slowly. Defining

$$\boldsymbol{\psi}_t \equiv \frac{\partial \boldsymbol{\theta}_t}{\partial g} \quad , \; \boldsymbol{M}_t \equiv \frac{\partial \boldsymbol{P}_t}{\partial g} \quad , \; \lambda'(g) \equiv \frac{d\lambda(g)}{dg} \tag{9}$$

Inserting Eq. 4 in Eq. 8 yields

$$\nabla_{g,t} = -E\left[\boldsymbol{x}_t^T \boldsymbol{\psi}_{t-1} \xi_t(\boldsymbol{\theta})\right] \tag{10}$$

In order to derive the Gauss Newton algorithm the second order derivative of $J_t(\boldsymbol{\theta})$ with respect to g is needed:

$$\begin{aligned} H_t &= \frac{\partial}{\partial g}\nabla_{g,t} = -\frac{\partial}{\partial g}E\left[\boldsymbol{x}_t^T\boldsymbol{\psi}_{t-1}\xi_t(\boldsymbol{\theta})\right] \\ &= E\left[(\boldsymbol{x}_t^T\boldsymbol{\psi}_{t-1})^2 - \boldsymbol{x}_t^T\frac{\partial\boldsymbol{\psi}_{t-1}}{\partial g}\xi_t(\boldsymbol{\theta})\right] \end{aligned} \tag{11}$$

A recursive estimate of $\boldsymbol{\psi}_{t-1}$ can be found using Eq. 14 below. $\frac{\partial\boldsymbol{\psi}_{t-1}}{\partial\lambda}$ in the second term only depends on information up to time $t-1$ and assuming that $\boldsymbol{\theta}$ is close to the true value, then ξ_t will be almost white noise (with zero mean and independent of the information set up to time $t-1$). Thus, the expectation of the second term is close to zero and the first term guarantees $H_t > 0$. A good approximation is therefore:

$$H_t = E\left[(\boldsymbol{x}_t^T\boldsymbol{\psi}_{t-1})^2\right] \tag{12}$$

For further details see Ljung and Söderström (1983). Simple exponential smoothing can be used to obtain a recursive estimate of H_t (Song et al., 2000):

$$H_t = (1-\alpha)H_{t-1} + \alpha(\boldsymbol{x}_t^T\boldsymbol{\psi}_{t-1})^2 \tag{13}$$

where α is the learning rate also used as stepsize when updating g_t (Eq. 17, below).

The update equation for $\boldsymbol{\psi}_t$ is found by differentiating Eq. 5, using $\boldsymbol{k}_t = \boldsymbol{P}_t\boldsymbol{x}_t$, which can be realized by using Eq. 6 and solving for $\boldsymbol{k}_t$ to get Eq. 6, and inserting Eq. 4:

$$\boldsymbol{\psi}_t = (\boldsymbol{I} - \boldsymbol{k}_t\boldsymbol{x}_t^T)\boldsymbol{\psi}_{t-1} + \boldsymbol{M}_t\boldsymbol{x}_t\xi_t \tag{14}$$

and similarly the update for $\boldsymbol{M}_t$ is found by differentiating Eq. 6:

$$\boldsymbol{M}_t = \frac{\lambda'}{\lambda}(\boldsymbol{k}_t\boldsymbol{k}_t^T - \boldsymbol{P}_t) + \frac{1}{\lambda}(\boldsymbol{I} - \boldsymbol{k}_t\boldsymbol{x}_t^T)\boldsymbol{M}_{t-1}(\boldsymbol{I} - \boldsymbol{k}_t\boldsymbol{x}_t^T)^T \tag{15}$$

And using:

$$\begin{aligned} &\frac{\partial}{\partial g}\left(\boldsymbol{k}_t\boldsymbol{x}_t^T\boldsymbol{P}_{t-1}\right) = \boldsymbol{k}_t\boldsymbol{x}_t^T\boldsymbol{M}_{t-1} + \frac{\partial\boldsymbol{k}_t}{\partial g}\boldsymbol{x}_t^T\boldsymbol{P}_{t-1} = \\ &\quad \boldsymbol{k}_t\boldsymbol{x}_t^T\boldsymbol{M}_{t-1} + \boldsymbol{M}_{t-1}\boldsymbol{x}_t\boldsymbol{k}_t^T - \lambda'\boldsymbol{k}_t\boldsymbol{k}_t^T - \boldsymbol{k}_t\boldsymbol{x}_t^T\boldsymbol{M}_{t-1}\boldsymbol{x}_t\boldsymbol{k}_t^T \end{aligned}$$

Approximating $\nabla_{g,t}$ by using the current estimate

$$\nabla_{g,t} = -\boldsymbol{x}_t^T\boldsymbol{\psi}_{t-1}\xi_t \tag{16}$$

then the steepest descent update of g_t yields

$$g_t = g_{t-1} + \alpha\boldsymbol{x}_t^T\boldsymbol{\psi}_{t-1}\xi_t \tag{17}$$

The proposed algorithm is given by using Eq. 2 in

$$\begin{aligned}
\boldsymbol{k}_t &= \frac{\boldsymbol{P}_{t-1}\boldsymbol{x}_t}{\lambda(g_{t-1}) + \boldsymbol{x}_t^T \boldsymbol{P}_{t-1}\boldsymbol{x}_t} && (18)\\
\xi_t &= y_t - \boldsymbol{x}_t^T \boldsymbol{\theta}_{t-1} && (19)\\
\boldsymbol{\theta}_t &= \boldsymbol{\theta}_{t-1} + \boldsymbol{k}_t \xi_t && (20)\\
\boldsymbol{P}_t &= \lambda^{-1}(g_{t-1})[\boldsymbol{I} - \boldsymbol{k}_t \boldsymbol{x}_t^T]\boldsymbol{P}_{t-1} && (21)\\
\boldsymbol{M}_t &= \lambda^{-1}(g_{t-1})[\boldsymbol{I} - \boldsymbol{k}_t \boldsymbol{x}_t^T]\boldsymbol{M}_{t-1}[\boldsymbol{I} - \boldsymbol{k}_t \boldsymbol{x}_t^T]^T + \\
&\quad \lambda^{-1}(g_{t-1})\lambda'(g_{t-1})\left(\boldsymbol{k}_t \boldsymbol{k}_t^T - \boldsymbol{P}_t\right) && (22)\\
g_t &= g_{t-1} + \alpha \boldsymbol{x}_t^T \boldsymbol{\psi}_{t-1}\xi_t && (23)\\
\boldsymbol{\psi}_t &= [\boldsymbol{I} - \boldsymbol{k}_t \boldsymbol{x}_t^T]\boldsymbol{\psi}_{t-1} + \boldsymbol{M}_t \boldsymbol{x}_t \xi_t && (24)
\end{aligned}$$

Note that $\lambda'(g)$ only appears in the update of $\boldsymbol{M}_t$.

The algorithm can be extended to a Gauss Newton algorithm by substituting Eq. 23 by:

$$\begin{aligned}
H_t &= (1-\alpha)H_{t-1} + \alpha(\boldsymbol{\psi}_{t-1}^T \boldsymbol{x}_t)^2 && (25)\\
g_t &= g_{t-1} + \alpha \boldsymbol{x}_t^T \boldsymbol{\psi}_{t-1}\xi_t / H_t && (26)
\end{aligned}$$

2.3 Simulation results

A simulation study was carried out to compare the stability of the steepest descent and Gauss-Newton variable forgetting factor algorithms using direct bounded update of λ_t and unbound optimization of g_t. The simulation model is given by: $y_t = b_t x_t + e_t$, where b_t is a time varying parameter given by: $b_t = 1.5 + 0.5\cos(2\pi f t)$ with $f = 10^{-4}$. x_t is autoregressive: $x_t = 0.975 x_{t-1} + 0.025 s_t$ with s_t i.i.d. uniformly distributed on $[1; 2]$. Finally, e_t is Gaussian i.i.d. noise with zero mean and standard deviation 0.7. This is a simple but very noisy system approximating the noise level when forecasting power production in wind farms. With the chosen very high noise level in combination with the chosen frequency in the change of the parameter a simple optimization of squared one step prediction errors on the last half of the data showed that 0.99 is the optimal fixed forgetting factor.

To illustrate the effect of introducing the unbounded optimization in the steepest descent algorithm we used the above simulation model to find near optimal α's for both versions, $\alpha_{bound} = 6 \cdot 10^{-6}$ and $\alpha_{unbound} = 0.5$, using $\lambda \in [0.6; 1]$ and $N_{min} = 3$, respectively. These settings were then used on a model with b_t being a constant equal to 1.5, and hence estimating the true model. A trace of the memory length for the two versions can be seen in Fig. 1. It's seen that the unbounded version makes an almost linear increase of the memory length whereas the version with bounds at unity and 0.6 has a very fluctuating memory, indeed many of the peaks reaches an infinite memory length ($\lambda = 1$). This unstable behavior is the result of not including the bounds in the optimization.

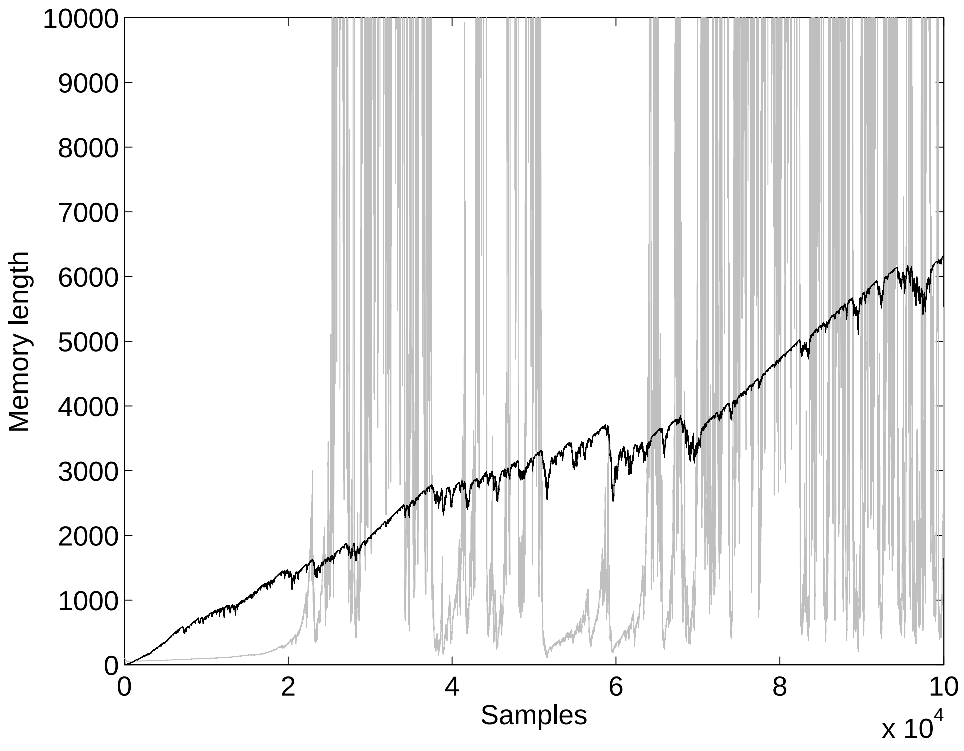

Figure 1: Comparing unbound (black) and bounded (gray) versions of the steepest descent algorithm on a simulation with a constant model using α's optimal for the sinusoid.

To investigate the effect of changing the initial conditions and the the length of the transient at different values of α the cumulative sum of squared one step prediction errors was calculated, see Fig. 2. Starting at the true parameter $(1.5 + 0.5 \times \cos(0) = 2)$ there is hardly any initial phase as the line is approximately linear from the beginning, this was independent of α within a wide range (See Fig. 3). If instead starting at a wrong value of the parameter the value of α determines how fast the forgetting factor can be changed. For large α's e.g. 10 the cumulative sum becomes linear quite fast but the slope is higher than for lower α's so it is not optimal. Using $\alpha = 10^{-6}$ is slightly better than 1 and the transient lasts less than 1500 samples in both cases.

Disregarding the initial transient (2000 samples in this case) the sum of squared one step prediction errors (SSPE) over a wide range of α's is shown in Fig. 3, this corresponds to the average slope in Fig. 2. For $\alpha \in [10^{-6}; 0.6]$ the SSPE is less than 1.015 times the sum of squared measurement errors (SSE $= \Sigma e_t^2$). And less than 1.010 times SSE in most cases. This should be compared with the optimal fixed forgetting factor of 0.99 resulting in SSPE 1.0083 times the SSE.

When using the Gauss Newton extension both in the bounded and unbounded setting the optimal α is about 10^{-4}. This value results in a very smooth Hessian. Hence it cannot adjust the Hessian when needed. Furthermore, a small α makes it more important to choose a good initial value of the Hessian. In this noisy setting it was decided not to use the Gauss Newton algorithm but

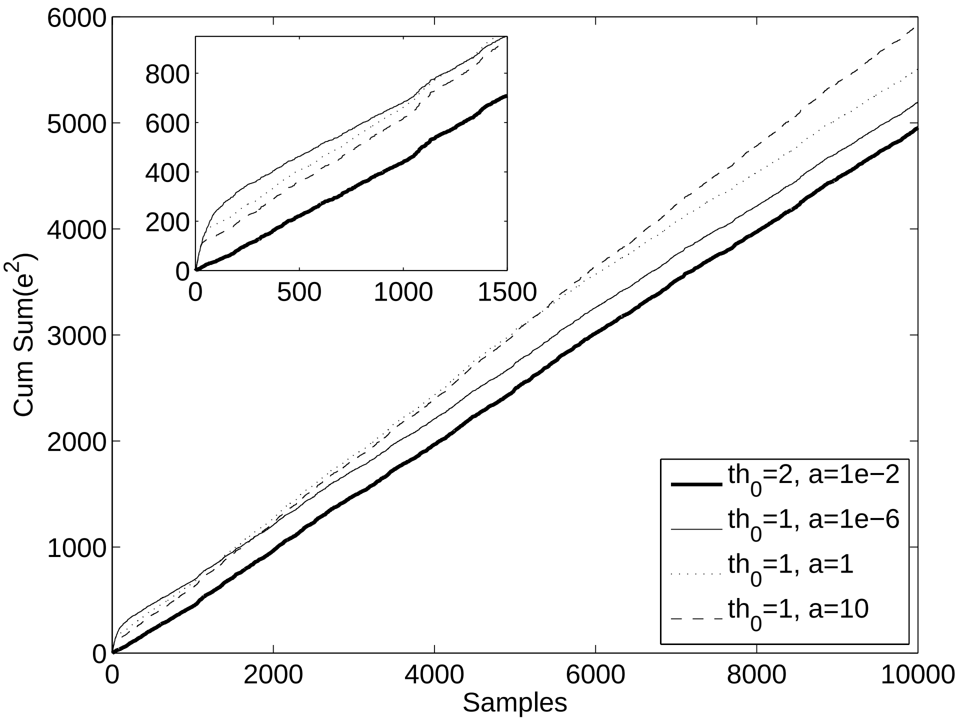

Figure 2: Comparing the cumulative sum of squared one step prediction errors for different initial settings (th_0) and different step lengths (a). The lines should be straight after a transient and the lower the slope the better the model.

may be appropriate for other applications.

2.4 Discussion

We find that the steepest descent algorithm in the unbounded setting is the better but both bounded and Gauss-Newton algorithms can all be tweaked to similar performance on the simple model used for illustration. The differences are seen when challenging the methods in different ways. The main motivation for this work was tuning a forgetting factor with an optimal value close to unity, i.e. using a model close to the true model. In such cases the upper bound will be hit when using the original formulation and the model will become somewhat unstable as was seen in Fig. 1. Unbounded optimization resulted in a smooth increase in the memory length.

It was expected that the Gauss-Newton algorithms would outperform the steepest descent algorithms. However, this were not observed, one reason is that α is both used to smoothen the estimate of the Hessian and as the step length in the update of λ. As the value of α has to be relatively low due to the high noise level the estimate of the Hessian is adjusted too late compared to the gradient estimates. Experiments with a different (larger) smoothing constant for the Hessian resulted in a more varying forgetting factor. However, it was not possible to identify a more optimal algorithm than just using the proposed unbounded steepest descent algorithm.

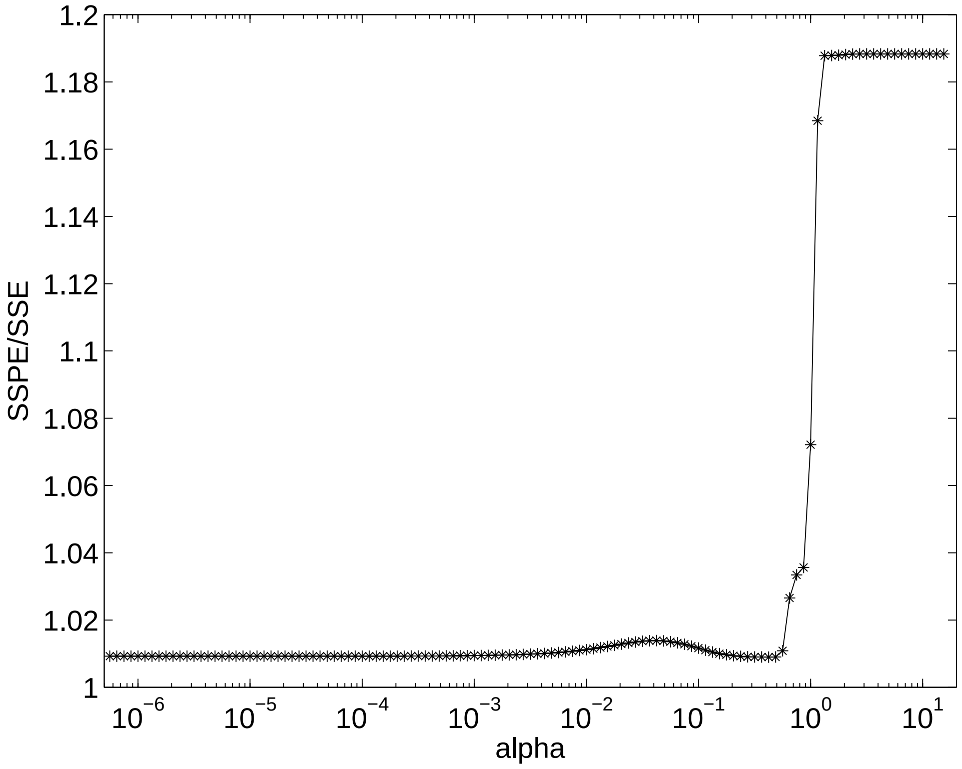

Figure 3: Changes in the sum of squared one step prediction errors (SSPE). Calculated removing a transient of 2,000 samples and normalized by the sum of squared measurement errors (SSE $= \Sigma e_t^2$).

In summary we find that when using a variable forgetting factor one should avoid hitting boundaries that are hidden for the optimizing scheme. One new solution reformulating the problem as an unbounded optimization has been presented and tested both using steepest descent and Gauss-Newton variable forgetting factor recursive least squares algorithms. Simulation results indicate that for noisy systems where the model used is close to the true model, as is the case for wind power predictions, steepest descent updates of an unbounded parameter is most suitable. On top α can be chosen within a wide range with only a small impact on performance.

3 RLS cond. par. model with adaptive bandwidth

3.1 Introduction

3.1.1 Background

When using local polynomial regression in a conditional parametric model a number of distinct points are set as fitting points for the local polynomials. The question addressed in this section is how to optimize the bandwidth at each of these fitting points. First a local formulation is used estimating the bandwidth at each point independent of all other points. Second a global approach where the bandwidth is given as a polynomial over the fitting points is outlined.

3.1.2 Framework

In the conditional parametric ARX-model (CPARX-model) with response y_s as presented by Nielsen et al. (2000) the explanatory variables are split in two groups. One group of variables $\mathbf{x}_s$ enter globally through coefficients depending on the other group of variables $\mathbf{u}_s$, i.e.

$$y_s = \mathbf{x}_s^T \boldsymbol{\theta}(\mathbf{u}_s) + e_s, \tag{27}$$

where $\boldsymbol{\theta}(\cdot)$ is a vector of coefficient-functions to be estimated and e_s is the noise term.

The functions $\boldsymbol{\theta}(\cdot)$ in (27) are estimated at a number of distinct points by approximating the functions using polynomials and fitting the resulting linear model locally to each of these *fitting points*. To be more specific let $\mathbf{u}$ denote a particular fitting point. Let $\theta_j(\cdot)$ be the j'th element of $\boldsymbol{\theta}(\cdot)$ and let $\mathbf{p}_{d(j)}(\mathbf{u})$ be a column vector of terms in the corresponding d-order polynomial evaluated at $\mathbf{u}$. The method is given by the following iterative algorithm.

$$\begin{aligned}
\boldsymbol{z}_t^T &= \left[x_{1,t}\boldsymbol{p}_{d(1)}^T(\boldsymbol{u}_t)\ldots x_{p,t}\boldsymbol{p}_{d(p)}^T(\boldsymbol{u}_t)\right] && (28)\\
\lambda_{eff,t}^{(i)} &= 1-(1-\lambda)W_{u^{(i)}}(\boldsymbol{u}_t) && (29)\\
\boldsymbol{\xi}_t &= y_t - \boldsymbol{z}_t^T\hat{\boldsymbol{\phi}}_{t-1}(\boldsymbol{u}^{(i)}) && (30)\\
R_{u^{(i)},t} &= \lambda_{eff,t}^{(i)}R_{u^{(i)},t-1} + W_{u^{(i)}}(\boldsymbol{u}_t)\boldsymbol{z}_t\boldsymbol{z}_t^T && (31)\\
\hat{\boldsymbol{\phi}}_t(\boldsymbol{u}^{(i)}) &= \hat{\boldsymbol{\phi}}_{t-1}(\boldsymbol{u}^{(i)}) + W_{u^{(i)}}(\boldsymbol{u}_t)R_{u^{(i)},t}^{-1}\boldsymbol{z}_t\boldsymbol{\xi}_t && (32)\\
\hat{\theta}_{jt}(\mathbf{u}^{(i)}) &= \mathbf{p}_{d(j)}^T(\mathbf{u}^{(i)})\,\hat{\boldsymbol{\phi}}_{j,t}(\mathbf{u}^{(i)});\;\; j=1,\ldots p && (33)
\end{aligned}$$

Where $W_{u^{(i)}}(\boldsymbol{u}_t)$ is the weight function used for fitting the local polynomials. Nielsen et al. used a tri-cube weight function (Defined as $(1-(\|\boldsymbol{u}_t-\boldsymbol{u}^{(i)}\|/h^{(i)})^3)^3$ if $\|\boldsymbol{u}_t-\boldsymbol{u}^{(i)}\| < h^{(i)}$ and zero otherwise). For further details and explanations see Nielsen et al. (2000).

In this section a generic weight function is used in the derivation and the tri-cube and Gaussian weight functions are used as examples. In the remaining part of this section the index $u^{(i)}$ indicating the fitting point has been omitted to simplify the expressions. Thus only considering one fitting point.

3.2 Local optimization of bandwidth

The idea is to optimize the bandwidth for each fitting point separately. And the bandwidth is to be optimized for each time step. It was chosen to use the expected weighted square of the one step prediction error:

$$J_t = \frac{1}{2}\,\mathrm{E}[W_t(\boldsymbol{u_t})\boldsymbol{\xi}_t^2(\boldsymbol{u_t})] \tag{34}$$

as the objective function. In order to make an unconstrained optimization of the bandwidth it was chosen to optimize g_t in:

$$h_t = \exp(g_t) \tag{35}$$

To do the optimization the derivative of the objective function with respect to g is needed:

$$\nabla_{g,t} = \frac{\partial J_t}{\partial g} = \mathrm{E}\left[\frac{1}{2}\frac{\partial W_t(\boldsymbol{u_t})}{\partial g}\boldsymbol{\xi}_t^2(\boldsymbol{u_t}) + W_t(\boldsymbol{u_t})\frac{\partial \boldsymbol{\xi}_t}{\partial g}\boldsymbol{\xi}_t\right] \tag{36}$$

In the following u is the fitting point for which the bandwidth is being optimized and u_t is the value at time t. The subscript u is omitted when writing ϕ and ψ. Defining

$$\boldsymbol{\psi}_t \equiv \frac{\partial \boldsymbol{\phi}_t}{\partial g} \quad , \quad \boldsymbol{M}_t \equiv \frac{\partial \boldsymbol{R}_t}{\partial g} \quad , \quad V_t(\boldsymbol{u_t}) \equiv \frac{\partial W_t(\boldsymbol{u_t})}{\partial g} \tag{37}$$

and using Eq. 30 the gradient can be written as:

$$\nabla_{g,t} = \mathrm{E}\left[\frac{1}{2}V_t(\boldsymbol{u_t})\boldsymbol{\xi}_t^2 - W_t(\boldsymbol{u_t})\boldsymbol{z}_t^T\hat{\boldsymbol{\psi}}_{t-1}\boldsymbol{\xi}_t\right] \tag{38}$$

A recursive estimate of $\boldsymbol{\psi}_t$ can be obtained by differentiation of Eq. 32:

$$\begin{aligned}\boldsymbol{\psi}_t \;=\; & \boldsymbol{\psi}_{t-1} + V_t(\boldsymbol{u_t})R_t^{-1}\boldsymbol{z}_t\boldsymbol{\xi}_t - W_t(\boldsymbol{u}_t)R_t^{-1}\boldsymbol{M}_tR_t^{-1}\boldsymbol{z}_t\boldsymbol{\xi}_t - \\ & W_t(\boldsymbol{u}_t)R_t^{-1}\boldsymbol{z}_t\boldsymbol{z}_t^T\hat{\boldsymbol{\psi}}_{t-1}\end{aligned} \tag{39}$$

Likewise, $\boldsymbol{M}_t$ can be estimated by differentiation of Eq. 31:

$$\boldsymbol{M}_t = \lambda_{eff,t}\boldsymbol{M}_{t-1} + V_t(\boldsymbol{u}_t)\boldsymbol{z}_t\boldsymbol{z}_t^T \tag{40}$$

What remains to make a steepest descent algorithm is the weight function and it's derivative.

3.2.1 Gauss-Newton optimization

A possible extension is to use second order derivatives of the objective function to do the optimization with a Gauss-Newton algorithm. Using the same notation as above:

$$\begin{aligned}
\nabla^2_{g,t} &= \frac{\partial^2 J_t}{\partial g^2} = \frac{\partial}{\partial g}\nabla_{g,t} \\
&= \frac{\partial}{\partial g}\,\mathrm{E}\left[\frac{1}{2}V_t(\boldsymbol{u_t})\boldsymbol{\xi}_t^2 - W_t(\boldsymbol{u_t})\boldsymbol{z}_t^T\hat{\boldsymbol{\psi}}_{t-1}\boldsymbol{\xi}_t\right] \\
&= \mathrm{E}\left[\frac{1}{2}\frac{\partial V_t(\boldsymbol{u_t})}{\partial g}\boldsymbol{\xi}_t^2 - 2V_t(\boldsymbol{u_t})\boldsymbol{z}_t^T\hat{\boldsymbol{\psi}}_{t-1}\boldsymbol{\xi}_t - \right. \\
&\qquad \left. W_t(\boldsymbol{u_t})\boldsymbol{z}_t^T\frac{\partial\hat{\boldsymbol{\psi}}_{t-1}}{\partial g}\boldsymbol{\xi}_t + W_t(\boldsymbol{u_t})\boldsymbol{z}_t^T\hat{\boldsymbol{\psi}}_{t-1}\boldsymbol{z}_t^T\hat{\boldsymbol{\psi}}_{t-1}\right]
\end{aligned} \tag{41}$$

It can be argued that close to the true set of parameters the expectation of the second and third terms are zero (See Ljung and Söderström (1983)).

Based on the experience with the Gauss-Newton approach for adjusting the forgetting factor it was decided to focus on the steepest descent algorithm.

3.2.2 Using Gaussian weight function

It was chosen to test the algorithm using a Gaussian kernel. It's easy to implement and has global support. Using Eq. 35 the Gaussian weight function is given by:

$$W_t(\boldsymbol{u_t}) = \frac{1}{\exp(g_t)\sqrt{2\pi}}\exp\left(-\frac{1}{2}\left(\frac{\|\boldsymbol{u}-\boldsymbol{u_t}\|}{\exp(g_t)}\right)^2\right) \tag{42}$$

Notice that pre-multiplying by the inverse bandwidth ensures that the integral is independent of the bandwidth. Besides the weight function the first order derivative with respect to g_t is needed:

$$V_t(\boldsymbol{u_t}) = \frac{\partial W_t(\boldsymbol{u_t})}{\partial g_t} = W_t(\boldsymbol{u_t})\left(\left(\frac{\|\boldsymbol{u}-\boldsymbol{u_t}\|}{\exp(g_t)}\right)^2 - 1\right) \tag{43}$$

Note that the derivative changes sign being positive for $\|\boldsymbol{u}-\boldsymbol{u_t}\| > \exp(g_t)$ and negative when closer to the fitting point.

The last part missing is an update of g_t and using the current estimate in Eq. 38 the steepest descent update is given by:

$$g_t = g_{t-1} - \alpha\left(\frac{1}{2}V_t(\boldsymbol{u_t})\boldsymbol{\xi}_t^2 - W_t(\boldsymbol{u_t})\boldsymbol{z}_t^T\hat{\boldsymbol{\psi}}_{t-1}\boldsymbol{\xi}_t\right) \tag{44}$$

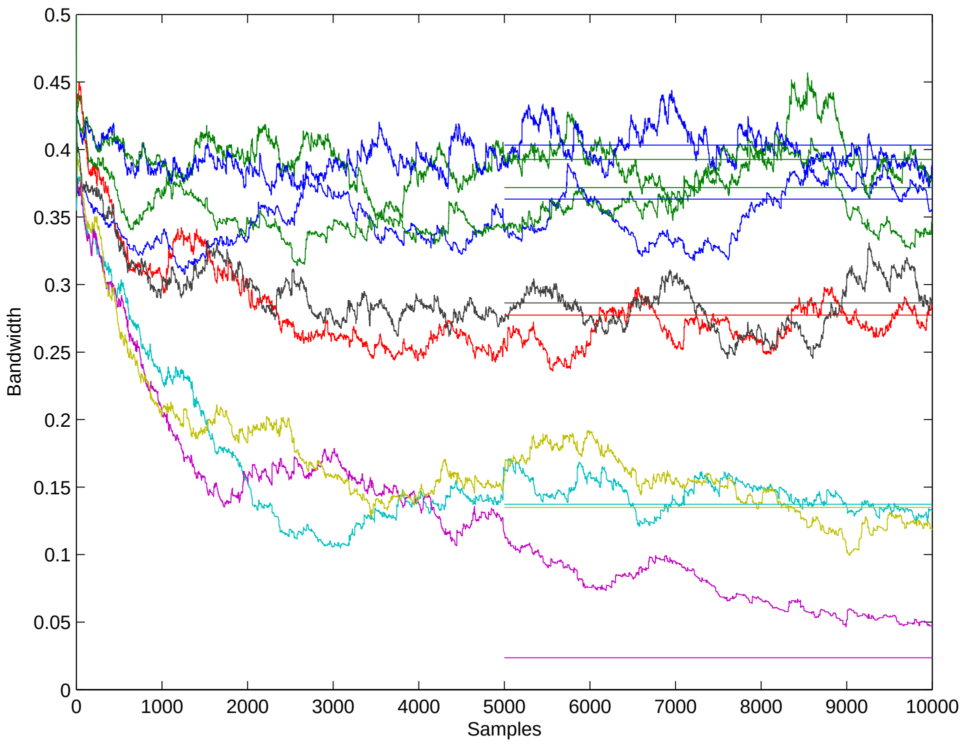

Figure 4: Using a Gaussian weight function to optimize the bandwidth at nine fitting points. Both steepest descent traces and fix bandwidth optimized on the last half of the data are shown. The 4 boundary points are blue and green, the central point is purple, the neighbour points of the central point are light blue and light green, and the remaining two points are black and red.

3.2.3 Example: Piecewise linear function

To test the ability to adjust the bandwith the following continuous piecewise linear function was used:

$$\theta(u) = \begin{cases} 1 & , 0 \leq u \leq 1 \\ u & , 1 < u \leq 2 \end{cases} \tag{45}$$

in combination with $y_t = x_t\theta(u_t) + e_t$, where $x_t \in U[1;2]$, $u_t \in U[0;2]$, and $e_t \in N(0, 0.25^2)$.

The trace of the bandwidth using the Gaussian weight function and a steepest descent update of the bandwidths individually for 9 fitting points distributed evenly from 0 to 2 and using local linear regression at each point can be seen in Fig. 4. On top of the nine traces of the bandwidth as optimized by steepest descent is corresponding lines showing the optimal fixed bandwidth measured over the last 5,000 samples. It is seen that the steepest descent does find the optimal value relatively fast when the initial value is not too far from the optimal value. The only trace that didn't converge within this timespan is the purple, corresponding to $u = 1$ where the change in slope is. That particular line is still converging after 10,000 samples so it should have been started at a more appropriate level if fast convergence was of interest, alternatively a larger α could have been chosen. Here the main focus was to show that it does converge towards

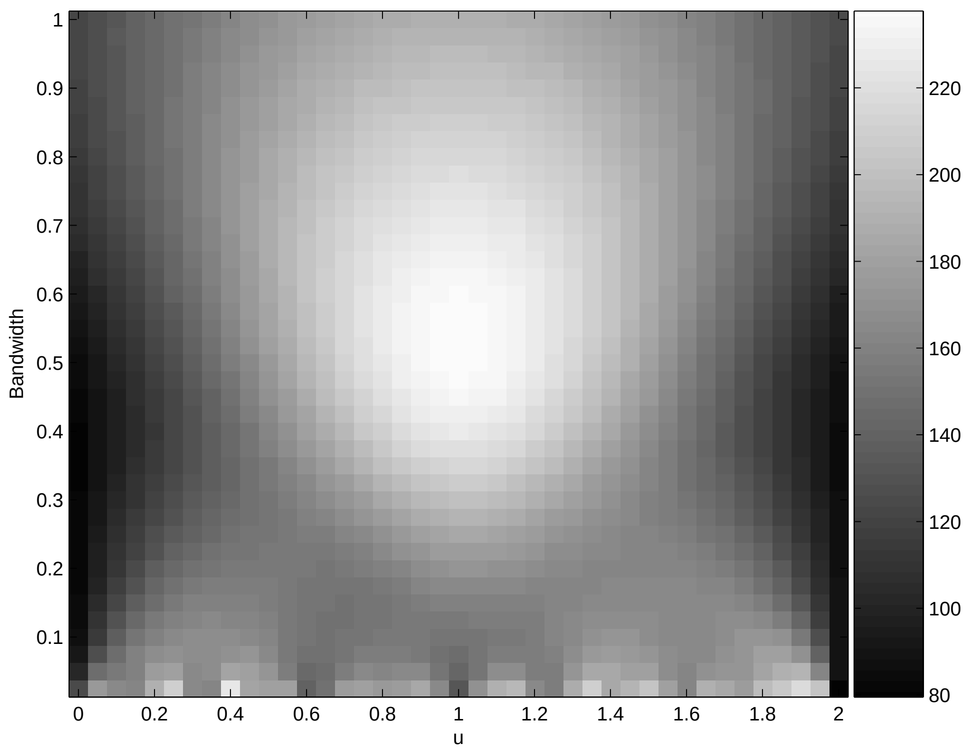

Figure 5: The objective function used for the lines in Fig. 4 for a range of fixed bandwidths and fitting points. The darker the lower value of the objective function.

the optimal value. Fig. 5 shows the value of the objective function as a function of the fitting point, u, and the bandwidth. The optimal bandwidth is the darkest cell in a vertical line over the chosen fitting point. The horizonal lines in Fig. 4 are examples of such optimal bandwidths. It is important to notice that if the initial bandwidth is set too high, e.g. above 0.6 for $u \in [0.6; 1.4]$, the algorithm will converge to a local minimum as it is taking small steps along the gradient.

3.2.4 Fitting points in higher dimensions

In the above the Euclidean distance was used in the weight function. Thus there should be no problems with a higher dimensional $\boldsymbol{u}$ as long as there is only one scaling parameter g_t. On the other hand there are cases where this does not hold, e.g. having wind direction and wind speed as the elements of $\boldsymbol{u}$. In those cases a product weight function should be used. Then there is a scaling parameter for each dimension ($\boldsymbol{g}_t = [g_{1,t}, g_{2,t}, \ldots]^T$) the dimension of $\boldsymbol{\psi}_t$, $\boldsymbol{M}_t$, and $\boldsymbol{V}_t$ is increased by one leading to a gradient vector rather than a scalar.

3.2.5 Using tri-cube weight function

In many cases it is preferable to use a weight function with non global support, i.e. only giving non zero weights to those points within the bandwidth from the fitting point. One such function is the tri-cube weight function:

$$W_t(\boldsymbol{u_t}) = \begin{cases} 0 & , \ n_t \geq 1 \\ \dfrac{140}{81\exp(g_t)}\left(1 - n_t^3\right)^3 & , \ n_t < 1 \end{cases} \tag{46}$$

Where $n_t = \|\boldsymbol{u} - \boldsymbol{u_t}\| / \exp(g_t)$ is the normalized distance to the fitting point. Again, it is important to notice that the weight function is normalized so that the integral is independent of the bandwidth. One motivation for the tri-cube weight function is that it has continuous zero, first, and second order derivatives and that having non global support reduces the computational burden in most settings.

Again the derivative is needed:

$$V_t(\boldsymbol{u_t}) = \begin{cases} 0 & , \ n_t \geq 1 \\ \dfrac{140}{81\exp(g_t)}\left(1 - n_t^3\right)^2\left(10n_t^3 - 1\right) & , \ n_t < 1 \end{cases} \tag{47}$$

and there is a change of sign as for the derivative of the Gaussian weight function. The two derivatives plotted as a function of the normalized distance can be seen in Fig. 6.

For comparison the same nine fitting points as used for the example with the Gaussian weight function was used. Fig. 7 shows the traces of the estimates of the bandwidth including horizontal lines over the last 5000 samples to show the optimal fixed values. Instead of using Eq. 35 a minimal bandwidth, h_0, was implemented as:

$$h_t = h_0 + \exp(g_t) \tag{48}$$

It was chosen to use $h_0 = 0.1$ and hence the optimal bandwidth of 0.05 for the purple line cannot be optained. The choice of h_0 corresponds to disallowing the lowest row in Fig. 8. In practice such a low bandwidth should not be used when the fitting points are as distant as in the present example. The weight functions of two neighboring fitting points should overlap, this can be obtained by increasing the number of fitting points or increasing the minimal bandwidth. When using the tri-cube weight function the optimal bandwidths are about three times as high as for the Gaussian weight function. Nevertheless the two behaves more or less the same as can also be seen in Fig. 8 (to be compared with Fig. 5) showing the sum of the weighted squares of one step prediction errors for fixed fitting point and bandwidth.

3.3 Discussion

The present section shows the derivation of a RLS based estimation of a conditional parametric model with variable bandwidth at each fitting point. A steepest descent approach was used to

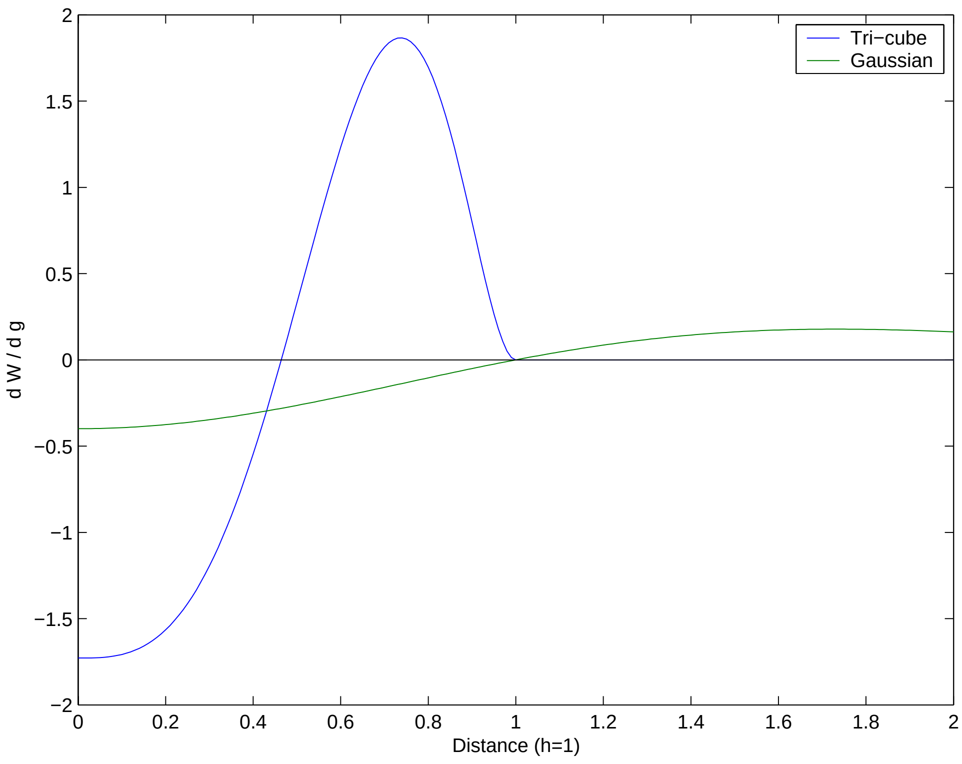

Figure 6: Comparing the derivative of the weight functions with respect to g.

optimize the bandwidth after each sample. An extension to using Gauss-Newton optimization has been suggested.

Both Gaussian and tri-cube weight functions have been put into this framework. The Gaussian is easy to implement and has global support which makes sure that all observations have a non zero weight and thus provides information irrespective of the bandwidth. The advantage of the tri-cube is that it does not have global support which reduces the computational burden. A lower bound on the bandwidth was needed to ensure numerical stability when using the tri-cube weight function but not when using the Gaussian weight function. The reason for this is probably due to the non-global versus global support. In most cases where predictions are of interest a lower bound should be considered based on the intra distance between the fitting points to assure a reasonable overlap of the weight functions.

4 Conclusion

It's been shown that it is feasible to make automatic tuning of the adaptiveness of tuning parameters in two classes of models. First for the forgetting factor of a recursive least squares (RLS) model and second for the bandwidth in a RLS based estimation of a conditional parametric

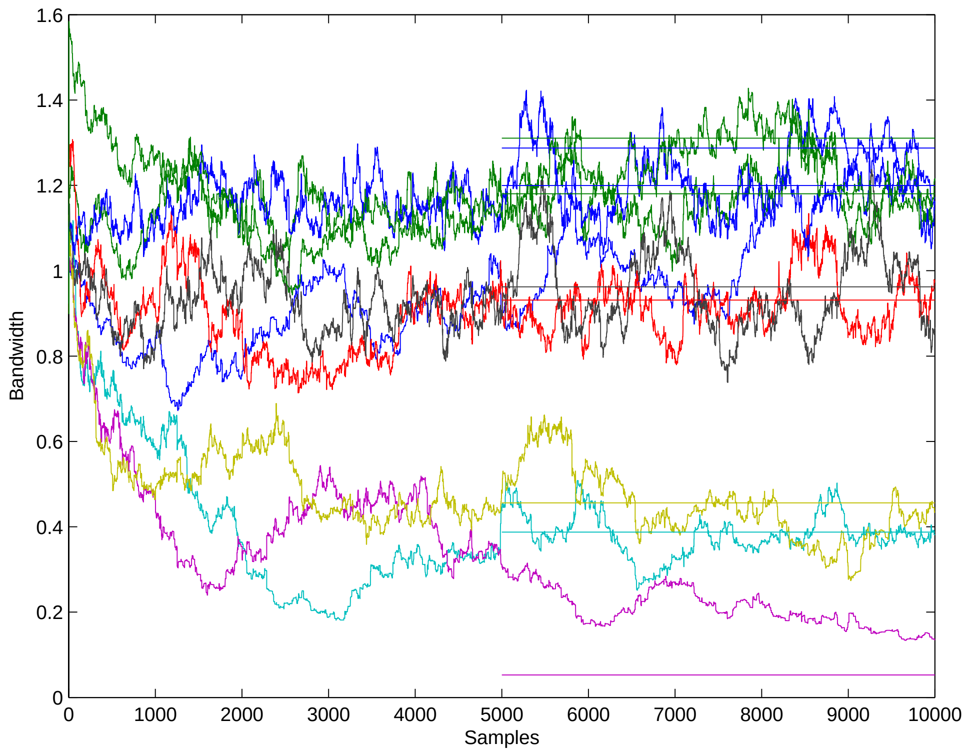

Figure 7: Using a tri-cube weight function to optimize the bandwidth at nine fitting points. Both steepest descent traces and fix bandwidth optimized on the last half of the data are shown.

model.

A discussion of the implementation in each of the two classes of models can be found by the end of the previous two sections.

Both classes have been tested using simulation studies representing common problems in numerical prediction of wind power production. It is suggested that further work should focus on higher dimensional properties of the suggested methods and inparticular on real life implementations of the algorithms.

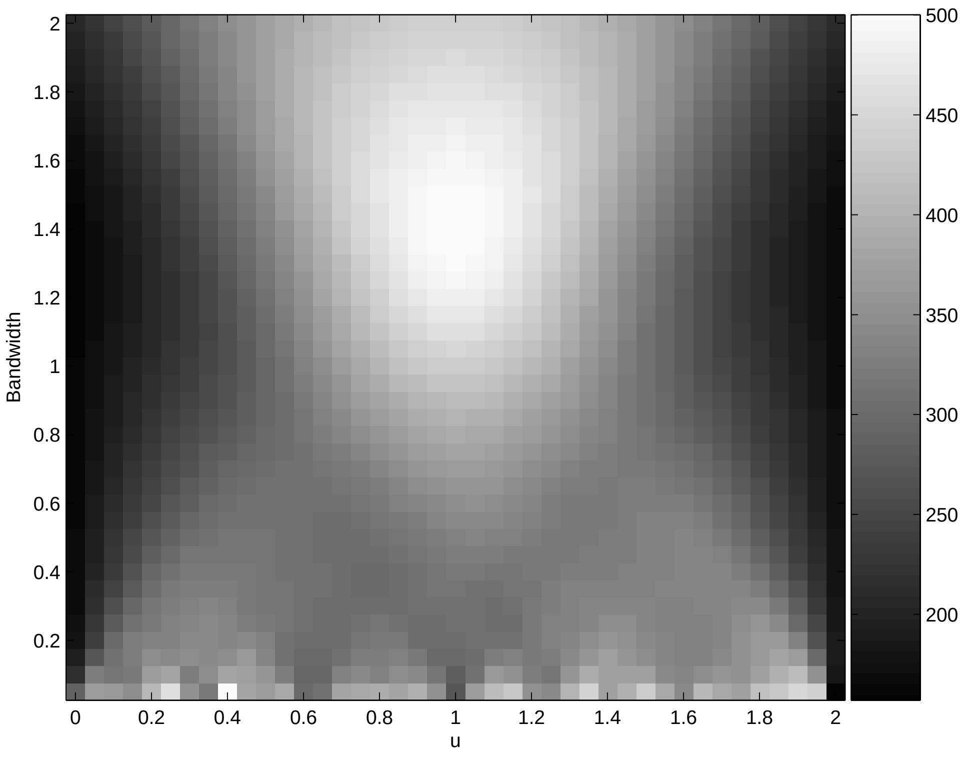

Figure 8: The objective function based on a tri-cube weight function which was used for the lines in Fig. 7 for a range of fixed bandwidths and fitting points. The darker the lower value of the objective function.

References

J. E. Cooper. On-line physical parameter estimation with adaptive forgetting factors. *Mechanical Systems and Signal Processing*, 14(5):705–730, 2000.

T. R. Fortescue, L. S. Kershenbaum, and B. E. Ydstie. Implementation of self-tuning regulators with variable forgetting factors. *Automatica*, 6:831–835, 1981.

S. Haykin. *Adaptive Filter Theory*. Prentice Hall, 3rd edition, 1996.

L. Ljung. *System Identification - Theory for the User*. Prentice Hall, 2nd edition, 1999.

L. Ljung and S. Gunnarsson. Adaption and tracking in system identification – a survey. *Automatica*, 26:7–22, 1990.

L. Ljung and T. Söderström. *Theory and Practice of Recursive Identification*. MIT Press, 1983.

Henrik Madsen, Henrik Aalborg Nielsen, and Torben Skov Nielsen. A tool for predicting the wind power production of off-shore wind plants. In *Proceedings of the Copenhagen Offshore Wind Conference & Exhibition*, Copenhagen, October 2005. Danish Wind Industry Association. `http://www.windpower.org/en/core.htm`.

M. B. Malik. State-space recursive least-squares with adaptive memory. In *Proc. ISPA03*, pages 146–151, 2003.

Henrik Aalborg Nielsen, Torben Skov Nielsen Alfred K. Joensen, Henrik Madsen, and Jan Holst. Tracking time-varying-coefficient functions. *International Journal of Adaptive Control and Signal Processing*, 14:813–828, 2000.

S. D. Peters and A. Antoniou. A parallel adaption algorithm for recursive-least-squares adaptive filters in nonstationary environments. *IEEE Transactions on signal processing*, 43(11):2484–2495, 1995.

C. F. So, S. C. Ng, and S. H. Leung. Gradient based variable forgetting factor RLS algorithm. *Signal Processing*, 83:1163–1175, 2003.

S. Song, J.-S. Lim, S. Baek, and K.-M. Sung. Gauss Newton variable forgetting factor recursive least squares for time varying parameter tracking. *Electronics Letters*, 36(11):988–990, 2000.