ESTIMATING THE PREDICTION UNCERTAINTY OF DYNAMIC NEURAL NETWORK PROCESS MODELS

K. Dadhe [*,1], **R. Gesthuisen** [*], **S. Engell** [*]

[*] *Process Control Laboratory,*
Department of Biochemical and Chemical Engineering,
University of Dortmund, D-44221 Dortmund, Germany,
Tel: +49/231/755-5127, Fax: +49/231/755-5129

Abstract: In this paper, the explicit calculation of prediction intervals for multi-step-ahead predictions from dynamic neural network models is described. Usually, asymptotic methods that are based on linearized descriptions of neural networks are applied with the potential problem of large coverage errors and too optimistic prediction intervals. The potential sources of these problems are the negligence of the network parameter uncertainties and the non-normality of error distributions. To overcome these restrictions, the bootstrap method is used here. The bootstrap is a tool from computational statistics. New formulations are introduced to apply the bootstrap to nonlinear time series models with exogenous inputs. The proposed method is illustrated by an analysis of a neural network model of a bioreactor benchmark problem.

Keywords: Prediction interval, neural networks, bootstrap, bioreactor.

1. INTRODUCTION

Chemical processes exhibit frequently strong nonlinearities and especially when operated in batch or semibatch mode or over a wide range cannot be controlled well by a linear controller which is designed around a nominal operation point. Hence, the application of nonlinear model predictive control (NMPC) schemes for chemical processes seems advantageous. A major drawback is the size and the complexity of rigoros dynamic models as the computational burden even with state of the art computers is prohibitive.

Nonlinear black box models may overcome this restriction and neural networks have gained considerably attention in both academic research as well as industrial applications. Their universal approximation abilities and the access to a wide range of software tools qualify them for the building of nonlinear dynamic black-box models which can be applied as prediction models in an NMPC scheme (see e.g. (Wang *et al.*, 2003)). In the application of this idea it turns out that a key issue is to estimate the prediction accuracy of the neural network models over medium to large prediction horizons.

However, a comparatively small number of articles in the literature deals with the assessment of neural network predictions. In the case of stationary mappings, several different approaches were reported. In (Chryssolouris *et al.*, 1996) a method based on linearization of the neural network is presented and in (Tibshirani, 1995) it is compared with two bootstrap methods.

The specific problem of the reliability or uncertainty of the neural network prediction within

[1] Corresponding author
E-mail: k.dadhe@bci.uni-dortmund.de

dynamic models has hardly been addressed systematically. On the other hand, neural network based predictive controller may show undesired behaviors when the process leaves the regime in which the prediction model had been trained. One possible solution is to divide the input space up into subspaces and to calculate the density of the training data in these subspaces. If the density is too low the neural network based NMPC switches to a conventional PID controller. This basic idea as well as technical considerations can be found in (Roßmann, 2002). A similar approach was described in (Tsai *et al.*, 2002) where a regional knowledge index is calculated. The index reflects the mean distance to all training patterns, and a supervising controller interpolates between the NMPC and a neural adaptive controller. Both approaches try to evaluate whether the current neural network input lies in a region with a high training data density or not. However, even if enough training data was used, the neural network predictions may be inaccurate, especially for growing prediction horizons.

In this paper, prediction intervals of multi-step-ahead forecasts of neural networks are calculated by bootstrap methods. A prediction interval is an upper and lower limit of the predicted value together with a probability α that future realizations of the process lie in between these bounds. For this purpose the residuals from the neural network training are taken as estimates of the true error distribution. The nominal neural network is then simulated many times with random draws from the error distribution. The newly generated data sets are used to estimate neural networks for each simulation run. The estimated error distribution represents the uncertainty of the network predictions with respect to noise or unmodeled process behavior, and the set of neural networks represents the uncertainty of the network parameters considering them as stochastic quantities as they are determined from noisy and limited data. The m-step-ahead prediction and its distribution are calculated from all neural network predictions and by adding an error term from the empirical error distribution during each prediction step.

The paper is structured as follows. Section 2 describes the basic methods used. In Section 3, extensions are made for dynamic models, Section 4 presents a linear method which is used in Section 5 where the results from a bioreactor simulation example are shown. The concluding Section 6 summarizes the presented results and gives an outlook to future work.

2. THE BOOTSTRAP

The bootstrap is a computer-based statistical method to estimate unknown distributions or parameters of distributions and can be separated into nonparametric or parametric techniques. The standard textbook which covers a wide range of applications is (Efron and Tibshirani, 1993). Basically, the bootstrap generates replicate pseudo-data sets by resampling from empirical distribution functions. The union of all bootstrap resamples is most likely to approximate the underlying true distribution. [2]

The standard nonparametric bootstrap method cannot be applied to dynamic models as it is based on the *iid*-assumption (independently and identically distributed) of the training data. As time series usually have serial correlations, the *iid*-assumption is obviously violated. In this case the bootstrap must be applied to the residuals of the training data rather than to the original data assuming that the residuals fulfill the *iid*-assumption.

A step-by-step guide for the calculation of prediction intervals without considerations of network uncertainties is as follows. The description is similar to that in (Clements and Taylor, 2001) and references therein even though only linear AR(p) models are considered there. The method will be denoted *Boot* in the sequel.

Step 1 The data is generated by the NAR(p) process

$$Y_k = f\left(\theta, Y_{k-1}, \ldots, Y_{k-p}\right) + E_k \qquad (1)$$

where E_k is a white noise process with distribution F_e, zero mean and finite second moment. θ is the parameter vector of the process. Simulate the process and get the time series $\mathcal{Y} = \{y_1, \ldots, y_{n+p}\}$ [3].

Step 2 Estimate the parameters of the neural network $\hat{\theta}$ and obtain the empirical distribution $F_{\hat{e}}$ [4] of the residuals

$$\hat{e}_k = y_k - f(\hat{\theta}, y_{k-1}, \ldots). \qquad (2)$$

If necessary center and rescale the residuals.

Step 3 Simulate B bootstrap continuations recursively for the m-step-ahead prediction

---

[2] This is the bootstrap paradigm. Statistical inference usually tries to find a description between a sample and the population the sample was drawn from. As the population and its parameters are generally unknown, the distribution of the (unknown) population is replaced by the (known) sample population and the original sample is replaced by the bootstrap one. A more formal definition is given in (Hall, 1988).

[3] Note the difference between the process Y_k and its realizations y_k.

[4] An empirical distribution function $F_{\hat{x}}$ assigns to each value of the sequence $\mathcal{X} = \{x_1, \ldots, x_n\}$ equal probabilities of n^{-1}. Sometimes n^{-1} is called mass.

$$y^{*b}_{k+j} = f\left(\hat{\theta}, y^{*b}_{k+j-1}, \ldots, y^{*b}_{k+j-p}\right) + e^{*b}_{k+j}, \quad (3)$$
$$j = 1, \ldots, m \quad b = 1, \ldots, B$$

and set $y^{*b}_{k+s} = y_{k+s}$ for $s \leq 0$ to condition the prediction interval on the last p observations which are supposed to be known. The e^{*b}_{k+j} are random draws from $F_{\hat{e}}$.

Step 4 Estimate the prediction intervals from the empirical distribution function of y^{*b}_{k+j}. Several methods may be applied, see (Hall, 1988) for a comprehensive treatment. Here, Efron's percentile method is used. Define the cumulative empirical distribution as

$$\alpha = G^*_B(h) = \frac{1}{B}\sum_{b=1}^{B} \mathcal{I}(y < h)$$

with the indicator function

$$\mathcal{I}(y < h) = \begin{cases} 1 & \text{if } y < h \\ 0 & \text{else} \end{cases},$$

the symmetric $100\alpha\%$-prediction interval is

$$\left[Q^*_B\left(\frac{1-\alpha}{2}\right), Q^*_B\left(\frac{1+\alpha}{2}\right)\right] \quad (4)$$

with $Q^*_B = G^{*-1}_B$.

This procedure does not account for the uncertainty in the neural network parameters. Extensions are presented to cope with this fact. The extended method is denoted *Boot-PU*.

Step 1-2 As above.

Step 3a Simulate B bootstrap time series $\mathcal{Y}^{*b}$ of length $n + p$ with the nominal model $f(\hat{\theta})$. Choose a random block of length p as initial values or fix them to $y_1, \ldots, y_p$.

Step 3b Estimate B neural networks with parameters $\hat{\theta}^{*b}$ for each $\mathcal{Y}^{*b}$.

Step 3c Simulate B bootstrap continuations recursively for the m-step-ahead prediction

$$y^{*b}_{k+j} = f\left(\hat{\theta}^{*b}, y^{*b}_{k+j-1}, \ldots, y^{*b}_{k+j-p}\right) + e^{*b}_{k+j}, \quad (5)$$
$$j = 1, \ldots, m \quad b = 1, \ldots, B$$

with the same settings as in eq. (3).

Step 4 As above.

Note that all B bootstrap models are trained in advance and that the potentially high number of neural networks does not rule out online applications as the evaluation of the network is comparatively fast in contrast to its training.

3. EXTENSIONS TO NARX MODELS

In the former section, the model class was restricted to nonlinear AR(p) models. The evaluation of the prediction uncertainty has gained considerable attention in the field of econometrics where exogenous inputs play a minor role. Control applications on the other hand demand for extensions to incorporate external inputs. Basically, there are three possible approaches. Assume that the model is of type

$$Y_k = f\left(\theta, Y_{k-1}, \ldots, Y_{k-p}, U_{k-1}, \ldots, U_{k-p}\right) + E_k. \quad (6)$$

Depending of the class of input signals, different strategies for resampling in Step 3a of the *Boot-PU* method can be applied. If the input signal is a deterministic one like sinusodial, chirp or rectangular, bootstrapping the empirical distribution function of the input sequence will produce signals with completely different spectral densities. As a consequence different process dynamics are excited and the estimated model most likely does not coincide with the nominal one. In this case it seems best to keep the input signal fixed or just shift it in phase. The resulting models and prediction intervals will thus be conditioned on the specific realization $\{u_k\}$ rather than the distribution F_u of the input signal.

For random signals with unknown distribution F_u, resampling from $\{u_1, \ldots, u_{n+p-1}\}$ will also generate bogus signals. A promising solution is the use of the empirical difference distribution $F_{\Delta\hat{u}}$. A bootstrap sample is most likely to have the same properties (e.g. spectral densities) as the original one and only rescaling of the bootstrap input signal might be necessary.

If the distribution of the input signals is known, it is obviously best to take random draws from the distribution function F_u. Step 3a is therefore extended as follows:

Step 3a, revised Simulate B bootstrap time series with the nominal model $f(\hat{\theta})$ to obtain $\mathcal{Y}^{*b}$ of length $n + p$.

$$y^{*b}_k = f\left(\hat{\theta}, y^{*b}_{k-1}, \ldots, y^{*b}_{k-p}, u^{*b}_{k-1}, \ldots, u^{*b}_{k-p}\right) + e^{*b}_k, \quad (7)$$
$$k = p+1, \ldots, p+n.$$

Choose a random block of length p as initial value or fix it to $\{y_1, \ldots, y_p, u_1, \ldots, u_p\}$. The e^{*b}_k is a random draw from $F_{\hat{e}}$. The input signals may be obtained by:

(1) Set $u^{*b}_k = u_{k-q^{*b}}$ to use the same input sequence as in $\mathcal{Y}$ with a potential (random) phase shift q^{*b}.

(2) Generate a difference data set $\Delta\mathcal{U} = \{u_2 - u_1, \ldots, u_{n+p-1} - u_{n+p-2}\}$ with corresponding distribution function $F_{\Delta\hat{u}}$. The input sequence will be $u^{*b}_k = u^{*b}_{k-1} + \Delta u^{*b}_k$ where Δu^{*b}_k is a random draw from $F_{\Delta\hat{u}}$. u^{*b}_1 can be zero or any value from $F_{\hat{u}}$. The bootstrap input sequence $\mathcal{U}^{*b}$ should be rescaled to have the same range as $\mathcal{U}$.

(3) If F_u is explicitly known, the input sequence $\{u_k^{*b}\}$ is just a realization of U.

4. LINEARIZATION-BASED APPROACH

The nominal neural network is linearized around an input vector $[y_{k-1}, \ldots, y_{k-p}, u_{k-1}, \ldots, u_{k-p}]$. This yields

$$Y_{k|k-1} = \delta + \phi_1 Y_{k-1} + \cdots + \phi_p Y_{k-p} + \psi_1 U_{k-1} + \cdots + \psi_p U_{k-p} + E_k \quad (8)$$

where $Y_{k|k-1}$ denotes the prediction obtained from a linearized model at instance $k-1$. As both input process U and parameters $\psi_{1\ldots p}$ and $\phi_{1\ldots p}$ are assumed to be known exactly, the approximated distribution of the m-step-ahead prediction is given by

$$Y_{k+j} \dot{\sim} \mathcal{N}(\hat{y}_{k+j} | \hat{\sigma}_e^2 + \sum_{i=1}^{p} \phi_i^2 \hat{\text{var}}(Y_{k+j-i})) \quad (9)$$

with $\hat{\text{var}}(Y_{k+j-i}) = 0$ for $j \leq i$ and variance $\hat{\sigma}_e^2$ estimated from the residuals between training data and neural network output. The resulting prediction interval is then

$$[\hat{y}_{k+j} - z_\alpha \hat{\text{se}}(Y_{k+j}); \hat{y}_{k+j} + z_\alpha \hat{\text{se}}(Y_{k+j})] \quad (10)$$

where z_α is the quantile of the normal distribution with $\Phi(z_\alpha) = \alpha$. The interval is symmetric about $\hat{y}_{k+j}$. The linearized method does not account for the uncertainty in the parameter estimates and for the possibly skewed distribution of Y_{k+j}.

5. SIMULATION EXAMPLE

5.1 *Model Description*

In this example, a continuous bioreactor benchmark is considered as described in (Puskorius and Feldkamp, 1994). The nonlinear coupled differential equations are given by

$$\dot{y}_1 = -y_1 u + y_1(1 - y_2) e^{y_2/\Gamma} \quad (11)$$

$$\dot{y}_2 = -y_2 u + y_1(1 - y_2) e^{y_2/\Gamma} \frac{1+\beta}{1+\beta-y_2} \quad (12)$$

where the state variables y_1 and y_2 represent the cell mass and nutrients conversion in dimensionless form. The exogenous variable u is the flow rate of nutrients into the constant volume tank and the nominal values of the parameters are $\Gamma = 0.48$ and $\beta = 0.02$. The process exhibits strong nonlinearities and has a tendency for instability and collapsing behavior ($y_{1,2} \to 0$ for constant u) for increasing flow rates. Fig. 1 shows the nonlinear behavior and reveals the necessity of a nonlinear model for e.g. model predictive control.

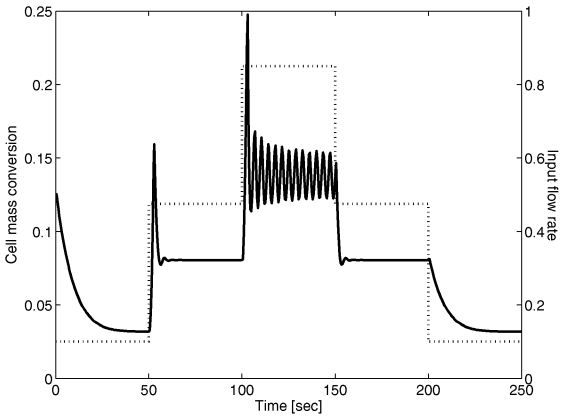

Fig. 1. Step responses of the bioreactor with initial values $y_{1,0} = 0.12$ and $y_{2,0} = 0.88$

5.2 *Neural Network Training*

The scope of this paper is not to find the best neural network structure and parameters for the problem of dynamic modeling of the bioreactor but to assess a neural network that has been obtained by standard methods.

(1) The rigoros process model is implemented in Matlab and simulated with the `ode23s`-integrator.
(2) The input signal for the training data set is an amplitude modulated pseudo random binary signal (APRBS) with a maximum flow of 0.85, a minimum of 0.1 and a switching time that is equally distributed over the interval $t_s \in [12.5s; 25s]$. The sampling interval is $0.5s$. For validation, the same type of signal is used with similar settings but different realizations.
(3) The feedforward neural network has one hidden layer with three neurons and sigmoidal activation functions. The order of the process (p) is four and the overall number of network parameters is 31. The process is simulated for 500 seconds which results in approximately 1000 training patterns. This is a relatively small number, in comparison (Puskorius and Feldkamp, 1994) use 10.000 points. An increase of generalization was obtained by the application of a ridge regression training procedure.
(4) Several networks with random initialization of the network parameters were trained and the one with the lowest root mean squared error on the training data set was selected.

Table 1 shows results for the validation data set. In the first column, the desired coverage probability α from eq. (4) is given, the second column denotes the method and the third one shows the actual coverage. An actual coverage of e. g. 88% means that 88% of the neural network five-step-ahead predictions for the validation data set lie

within the calculated prediction intervals. If the methods for calculating the prediction intervals are (asymptotically) correct, then for large data sets and large bootstrap samples the actual coverage tends to the nominal one.

$$\lim_{n\to\infty} \lim_{B\to\infty} \text{Prob}\left[y_{\text{true}} \in \text{eq. (4)}\right] = \alpha \qquad (13)$$

The fourth column shows the width of the prediction intervals averaged over all validation data points. The standard deviation of the width is given in brackets ($\hat{\sigma}$).

Table 1. Coverage for five-step-ahead prediction for Validation Data Set.

Nominal Coverage	Method	Actual Coverage	Average Width ($\hat{\sigma}$)
50%	Linear	51.05%	0.52 (3.89)
	Boot	51.65%	0.01 (0.01)
	Boot-PU	58.54%	0.01 (0.03)
80%	Linear	64.34%	0.99 (7.38)
	Boot	81.12%	0.02 (0.04)
	Boot-PU	85.41%	0.02 (0.06)
90%	Linear	65.63%	1.27 (9.48)
	Boot	88.11%	0.03 (0.05)
	Boot-PU	90.61%	0.04 (0.08)
95%	Linear	67.13%	1.51 (11.30)
	Boot	92.11%	0.04 (0.07)
	Boot-PU	94.61%	0.05 (0.10)
99%	Linear	70.83%	1.98 (14.84)
	Boot	96.70%	0.07 (0.10)
	Boot-PU	97.70%	0.08 (0.17)

The method based on linearization constantly underestimates the prediction interval leading to far too optimistic results. The poor coverage and the relatively wide interval width originates from the fact that only symmetric intervals can be calculated due to the normality assumption. The large standard error $\hat{\sigma}$ is a consequence of the linearization as the estimated variance is rather high when step changes on the input variable occur and the nonlinearity of the process has a significant effect on the prediction. The bootstrap method which only uses random draws form the empirical residual distribution $F_{\hat{e}}$ for the prediction (denoted *Boot*) and therefore conditions them on the nominal network parameters $\hat{\theta}$ shows much better coverage than the linear method with the advantage of small average interval widths. It is only outperformed by the bootstrap method which accounts for the parameter uncertainty (hence denoted *Boot-PU*) which shows very good coverage with a slight tendency to overestimate the prediction intervals for low nominal coverage.

To illustrate the empirical distribution functions and the variation of the prediction intervals with time, the five-step-ahead prediction is compared with the plant behavior. Fig. 2 shows about 15% of the validation data set. The neural network shows good approximation abilities of the true process behavior. As the order of the NARX process is four, the five-step-ahead prediction $y_{k+5|k}$ is calculated completely from former neural network predictions because the values $y_{k+4|k}, \ldots, y_{k+1|k}$ are not known at time instance k. Only the large step at $t = 10$ sec causes a large error of the neural network prediction.

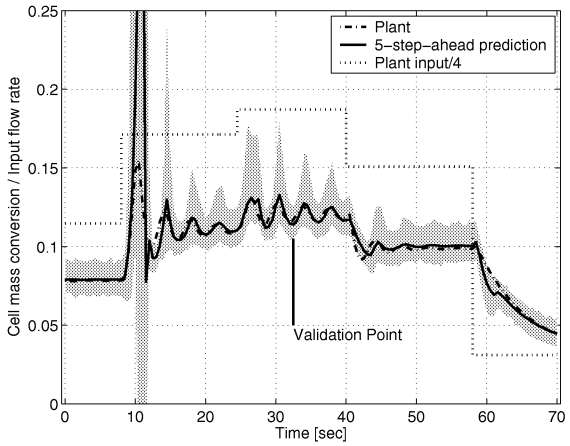

Fig. 2. Plant step response and neural network five-step-ahead prediction. The area shaded in gray is the 95% prediction interval of the *Boot-PU* method.

The distribution of the bootstrap predictions, the corresponding 95% confidence intervals as well as the normal approximations for the point indicated in Fig. 2 as validation point are shown in Fig. 3. The non-stationarity of the process at this point causes the linear method to estimate a large prediction uncertainty, whereas the neural network exhibits a sufficient performance there. The bootstrap distribution of the five-step-ahead prediction is slightly skewed to larger values of the predicted variable indicating that the nominal neural network may underestimate the true value. In comparison to the linear method, the width of the prediction interval is rather small. Both bootstrap methods yield almost equal intervals here.

Another important aspect is the asymptotic behavior and the rate of convergence to the limiting distribution. Table 2 shows the actual coverage for different nominal coverage with respect to an increasing number of bootstrap predictions B. The use of more than 400 bootstrap predictions does not increase the coverage accuracy significantly.

6. CONCLUSION

In this paper, a method has been introduced to evaluate the prediction uncertainty of neural net-

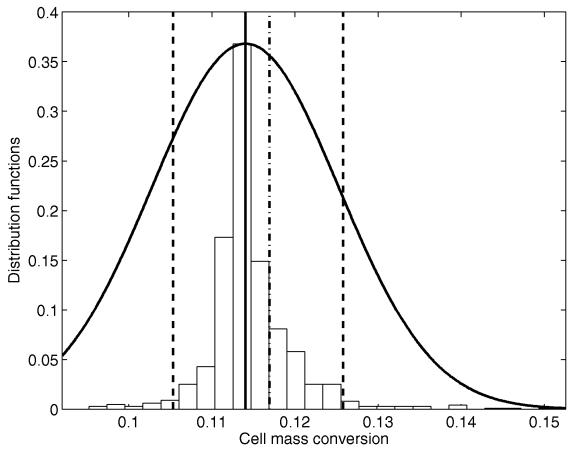

Fig. 3. Distribution functions at the point indicated in Fig. 2. The bars show the empirical distribution of the bootstrap predictions, the solid vertical line is the neural network prediction, the dash-dotted line is the real process output, the dashed lines are the 95% bootstrap intervals and the bell-shaped curve (normalized to the maximal bootstrap value) is the normal approximation of the error distribution that results from the linearization based method.

Table 2. Influence of the number of bootstrap samples

	Validation Data Set			
	Nominal coverage 80%		Nominal coverage 95%	
B	Boot	Boot-PU	Boot	Boot-PU
20	78.3%	82.1%	85.3%	89.0%
100	80.2%	83.5%	90.7%	92.8%
400	81.0%	85.1%	92.0%	94.6%
1000	81.1%	85.4%	92.1%	94.6%

works for modeling of nonlinear dynamic systems. This aspect has not yet attracted the necessary attention even though unreliable predictions can lead to erratic behavior of e.g. nonlinear model predictive controllers.

The bootstrap as a tool from computational statistics is used to circumvent the limitations that are imposed by the nonlinearity of the applied neural network models and by the normality assumption of the error distributions. The innovation improves the coverage largely and narrows the width of the prediction intervals as nonsymmetric intervals are possible.

Although the proposed method gives additional insight into the prediction uncertainty of neural networks, further research is necessary to improve the estimation of prediction intervals. Using eq. (1) tacitly assumes that the data generating process lies in the class of neural networks used for modeling. If this assumption is not justified, the effects on the prediction intervals have to be examined as the empirical error distribution is potentially biased. Another aspect is the unconditionality of the empirical error distribution which implies that the variance of the error process does not depend on the current neural network input. If the neural network operates in ranges with low training data, sampling from the empirical error distribution might be too optimistic and if it operates in a range with high training data density sampling from the same distribution might be too pessimistic. Conditioning the empirical error distribution on the current point of operation will improve both the actual coverage of a complete data set and reduce the conservativeness of the empirical distribution of the multi-step-ahead prediction errors.

7. ACKNOWLEDGEMENTS

Part of this work was sponsored by the DFG (Deutsche Forschungsgemeinschaft).

8. REFERENCES

Chryssolouris, G., M. Lee and A. Ramsey (1996). Confidence interval prediction for neural network models. *IEEE Trans. Neural Networks* **7**(1), 229–232.

Clements, M. P. and N. Taylor (2001). Bootstrapping prediction intervals for autoregressive models. *International Journal of Forecasting* **17**, 247–267.

Efron, B and R. J. Tibshirani (1993). *An Introduction to the Bootstrap*. Chapman & Hall.

Hall, P. (1988). Theoretical comparison of bootstrap confidence intervals. *The Annals of Statistics* **16**(3), 927–953.

Puskorius, G. V. and L. A. Feldkamp (1994). Neurocontrol of nonlinear dynamical systems with kalman filter trained recurrent networks. *IEEE Trans. Neural Networks* **5**(2), 279–297.

Roßmann, V. (2002). Prädiktive Regelung mit Neuronalen Netzen (Predictive control with neural networks). PhD thesis. University of Dortmund, Department of Biochemical and Chemical Engineering.

Tibshirani, R. (1995). A comparison of some error estimates for neural networks. *Neural Computation* **8**, 152–163.

Tsai, P. F., J. Z. Chu, S. S. Jang and S. S. Shieh (2002). Developing a robust model predictive control architecture through regional knowledge analysis of artificial neural networks. *J. Proc. Contr.* **13**, 423–435.

Wang, C., K.-U. Klatt, G. Dünnebier, S. Engell and F. Hanisch (2003). Neural network-based identification of SMB chromatographic processes. *Control Engineering Practice* **11**(8), 949–959.