ASSESSING THE PREDICTIONS OF DYNAMIC NEURAL NETWORKS

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

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

Abstract: In this paper, the estimation of prediction intervals for multi-step-ahead predictions from dynamic neural network models is described. Usually, asymptotic methods based on linearizations 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 the error distribution. To overcome these restrictions, bootstrap methods are used here. New formulations are introduced to apply the bootstrap to nonlinear time series models with exogenous input. An explicit model of the error process considers the influence of different training data densities on the empirical error distribution. A Monte Carlo study illustrates the proposed methods.

Keywords: Dynamic Neural Networks, Prediction, Bootstrapping

1. INTRODUCTION

Many chemical and biochemical processes exhibit strong nonlinearities. First principles modeling based on physical and chemical laws is difficult and time consuming, and the resulting models are often of large size. In control applications, data-based models are therefore often preferred. Linear models are easy to estimate but often not sufficiently accurate.

Nonlinear black box models may overcome this problem. Neural networks have gained considerable 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. For applications of this idea it turned 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 (Rivals and Personnaz, 2000) (and references therein) methods based on linearizations of the neural network are presented and in (Tibshirani, 1995) they are compared with two bootstrap methods.

The problem of the reliability of dynamic neural network predictions after the training and validation procedures have been completed has hardly been addressed systematically. 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 into subspaces and to calculate the density of the training data in these subspaces and to use the models only where enough data was avail-

[1] Corresponding author
E-mail: s.engell@bci.uni-dortmund.de

able, see (Roßmann, 2002). A similar approach was described in (Tsai *et al.*, 2002) where a regional knowledge index is calculated. If the measures indicate poor neural network predictions, the neural network based NMPC switches to a conventional controller. Both approaches try to evaluate whether the current neural network input lies in a region with high training data density or not. However, even if enough training data was used, the neural network predictions may be inaccurate, especially for larger prediction horizons.

In this paper, prediction intervals of multi-step-ahead forecasts of neural networks are calculated by bootstrap methods. The proposed methods yield prediction intervals that can be used as reliability measures to assess multi-step-ahead predictions within an NMPC framework. A prediction interval is an upper and lower limit of the predicted value together with a probability γ that future realizations of the process lie between these bounds. For this purpose, the residuals from the neural network training are taken as estimates of the true error distribution.

The residuals are mainly caused by two effects. One source is noise in the training data and the second source is the mismatch between the neural network and the data generating process. The mismatch is likely to be worse in regions where the density of the training data is low. (Bishop, 1995) showed that the variance of the neural network parameter estimates is high in regions with low training data density. As the output variance obeys (at least locally) the Gaussian error propagation, it is high in these regions, too. Hence, the neural network model has poor approximation and prediction capabilities in these regions and the error distribution depends on the current input and state. An explicit description of the error process is realized by a GARCH model to account for varying error variances. GARCH (Generalized Autoregressive Conditional Heteroskedasticity) models were introduced in (Bollerslev, 1986) to explain time series from economic processes.

The nominal neural network and the GARCH model are simulated many times with random draws from the resulting error distributions. The generated bootstrap data sets are used to estimate bootstrap neural networks and bootstrap error models for each simulation run. The estimated error distribution represents the uncertainty of the network predictions with respect to noise and unmodeled process behavior, and the set of bootstrap 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 predictions and their distributions are calculated from the nominal and the bootstrap neural network predictions, adding a random draw from the empirical error distribution during each prediction step.

This paper introduces the application of the GARCH error model and performs a more extensive simulation study compared to the preceding paper (Dadhe *et al.*, 2004) considering different non-normal error processes, different methods to estimate the resulting prediction intervals and a further application of the bootstrap method to dynamic neural networks. The paper is structured as follows. Section 2 gives an introduction to the bootstrap methods used here with a focus on nonlinear dynamic models, and explains the application of the GARCH for modeling the conditional variances. In Section 3, extensions are made for dynamic models with exogenous inputs typical for control applications. In Section 4, the Monte Carlo results from a neural network simulation example are shown. The concluding Section 5 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. It can be classified 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 most likely approximates the underlying true distribution. The standard nonparametric bootstrap method cannot be applied to dynamic models as it is based on the *iid*-assumption (independent 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 given below. The description is similar to that in (Clements and Taylor, 2001) and (Pascual *et al.*, 2001), however only linear AR(p) and ARIMA(p, d, q) models are considered there. The method will be denoted *C-B* (*conditional bootstrap*) 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 the noise process with distribution function F_e, zero mean and finite second moment. θ is the parameter vector of the process. Simulate the process and get the time series $\mathcal{Y} = \{y_1, \ldots, y_n\}$ from the initial values $\{y_{-p+1}, \ldots, y_0\}$.

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

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

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

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

and set $y^{*b}_{k+s} = y_{k+s}$ for $s \le 0$ to condition the predictions 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. The cumulative empirical distribution is defined as

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

with the indicator function

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

Then the symmetric 100γ%-prediction interval is

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

with $Q^*_B = G^{*-1}_B$. (Hall, 1988) proposes to use quantiles of the normalized bootstrap distribution instead, yielding prediction intervals

$$\left[\hat{y}_{k+j} - Q^*_{B,H}\left(\frac{1+\gamma}{2}\right), \hat{y}_{k+j} - Q^*_{B,H}\left(\frac{1-\gamma}{2}\right)\right]. \tag{5}$$

$Q^*_{B,H}((1-\gamma)/2)$ and $Q^*_{B,H}((1+\gamma)/2)$ are the respective quantiles of the bootstrap distribution of $Y^{*b}_{k+j} - \hat{y}_{k+j}$.

The *C-B* procedure does not account for the uncertainty in the neural network parameters and therefore conditions the prediction intervals on one specific neural network with the parameter $\hat{\theta}$. As the parameters are determined from limited and noisy data, they should be considered as stochastic quantities, too. Extensions are presented below to cope with this fact.

In (Dadhe *et al.*, 2004) it was stressed that the variance of the error distribution varies with the input of the neural network. *Step 2* in the previously listed method does not condition the error distribution on the current state (input and output) of the neural network. In regions with high training data density the error variance is probably smaller than in regions where the network has only limited information. It is therefore questionable to assume an error process E_k with constant variance. Statistical tests can prove whether E_k is homoskedastic (constant variance) and whether the consideration of an explicit error model to account for varying variance is necessary.

Here, two extended methods are applied. *PU-B* (*parameter uncertainty bootstrap*) considers uncertainties in the neural network parameter estimates and *GARCH-B* (*GARCH bootstrap*) additionally accounts for non-constant error variances with a (bootstrapped) error model. The bootstrap procedure for the GARCH model is taken from (Pascual *et al.*, 2000).

Step 1 and 4. As above.

Step 2. Obtain the parameter estimate $\hat{\theta}$ for the neural network. Calculate the residuals

$$\hat{e}_k = y_k - f\left(\hat{\theta}, y_{k-1}, \ldots\right)$$

and use the statistical tests described in (Bollerslev, 1986) to check whether the residuals $\{\hat{e}_k\}$ show GARCH behaviour. Define the GARCH(1,1) model

$$\hat{e}_k = \sqrt{\hat{h}_k}\hat{\nu}_k \tag{6}$$
$$\hat{h}_k = \omega + \alpha\hat{e}^2_{k-1} + \beta\hat{h}_{k-1}. \tag{7}$$

The error model parameters $\{\omega, \alpha, \beta\}$ are determined by minimization of the (negative) log-likelihood function. Even though it would be more precise to solve the log-likelihood for both the neural network and GARCH model together, the sequential procedure bears the advantage that standard software tools for both model types can be used. The empirical normalized error distribution $F_{\hat{\nu}}$ is

$$F_{\hat{\nu}} = \{\hat{\nu}_k\} = \left\{\frac{\hat{e}_k}{\sqrt{\hat{h}_k}}\right\}\Bigg|_{\hat{\omega},\hat{\alpha},\hat{\beta}} \tag{8}$$

Step 3a. Simulate B bootstrap time series $\mathcal{Y}^{*b}$, $b = 1, \ldots, B$ of length n with the nominal models $f(\hat{\theta})$ and $\{\hat{\omega}, \hat{\alpha}, \hat{\beta}\}$. Choose a random block of length p as initial values or fix them to $y_{-p+1}, \ldots, y_0$, e^2_0 and h_0. The bootstrap GARCH series are generated according to

$$h^{*b}_k = \hat{\omega} + \hat{\alpha}e^{*b\,2}_{k-1} + \hat{\beta}h^{*b}_{k-1} \tag{9}$$
$$e^{*b}_k = \sqrt{h^{*b}_k}\nu^{*b}_k, \tag{10}$$

where the ν^{*b}_k are random draws from $F_{\hat{\nu}}$.

Step 3b. Estimate B neural networks with parameters $\hat{\theta}^{*b}$ and B GARCH model parameters $\{\hat{\omega}^{*b}, \hat{\alpha}^{*b}, \hat{\beta}^{*b}\}$ for each $\mathcal{Y}^{*b}$.

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

$$h^{*b}_{k+j} = \hat{\omega}^{*b} + \hat{\alpha}^{*b} e^{*b\ 2}_{k+j-1} + \hat{\beta}^{*b} h^{*b}_{k+j-1} \tag{11}$$

$$e^{*b}_{k+j} = \sqrt{h^{*b}_{k+j}} \nu^{*b}_{k+j} \tag{12}$$

$$y^{*b}_{k+j} = f\left(\hat{\theta}^{*b}, y^{*b}_{k+j-1}, \ldots, y^{*b}_{k+j-p}\right) + e^{*b}_{k+j}, \tag{13}$$

$$j = 1, \ldots, m \quad b = 1, \ldots, B$$

with the same settings as in eq. (3). The conditional variance at time instance k is estimated following (Pascual *et al.*, 2000) as

$$h^{*b}_k = h^{*b}_0 + \hat{\alpha}^{*b} \sum_{\ell=0}^{k-2} \hat{\beta}^{*b\ \ell} \left(e^2_{k-\ell-1} - h^{*b}_0\right) \tag{14}$$

with the unconditional variance

$$h^{*b}_0 = \frac{\hat{\omega}^{*b}}{1 - \hat{\alpha}^{*b} - \hat{\beta}^{*b}}. \tag{15}$$

By replacing $F_{\hat{\nu}}$ by $F_{\hat{e}}$ and omitting the GARCH model the above *GARCH-B* method reduces to the *PU-B* method. Note that all B bootstrap models are trained and estimated in advance and that the potentially high number of neural networks does not rule out online applications as the evaluation of the networks is comparatively fast in contrast to their training.

3. EXTENSIONS TO NARX MODELS

In the former section, the model class was restricted to nonlinear AR(p) models. 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

$$Y_k = f(\theta, Y_{k-1}, \ldots, Y_{k-p}, U_{k-1}, \ldots, U_{k-p}) + E_k. \tag{16}$$

For a simpler notation, equal lags p for the input and output variables are assumed. Depending on the class of input signals, different strategies for resampling in *Step 3a* of the *GARCH-B* or *PU-B* methods 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 $\{u_1, \ldots, u_n\}$ will also generate bogus signals. A promising solution is the use of the empirical difference distribution $F_{\Delta\hat{u}}$. A bootstrap sample drawn from $F_{\Delta\hat{u}}$ is most likely to have similar properties (e.g. spectral densities) as the original one. Rescaling of the bootstrap input signal might be necessary to ensure that the bootstrap input series lies in the same range as the realization $\{u_k\}$.

If the distribution of the input signals is known, obviously the best approach is 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 $f(\hat{\theta})$ and $\{\hat{\omega}, \hat{\alpha}, \hat{\beta}\}$ to obtain $\mathcal{Y}^{*b}$ of length n. The h^{*b}_{k+j} and e^{*b}_{k+j} are taken from eq. (11-12).

$$y^{*b}_k = f\left(\hat{\theta}, y^{*b}_{k-1}, \ldots, y^{*b}_{k-p}, u^{*b}_{k-1}, \ldots, u^{*b}_{k-p}\right) + e^{*b}_k, \quad k = 1, \ldots, n. \tag{17}$$

Choose a random block of length p as initial value or fix it to $\{y_{-p+1}, \ldots, y_0, u_{-p+1}, \ldots, u_0\}$. 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 $\Delta\mathcal{U} = \{u_1 - u_0, \ldots, u_{n-1} - u_{n-2}\}$ 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}_0 can be zero, any value from $F_{\hat{u}}$ or u_0. 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 $\{u^{*b}_k\}$ is just a realization of U.

4. SIMULATION EXAMPLE

4.1 *Model Description*

The data generating process is a nonlinear AR(2) model with exogenous input u originally trained with data obtained from simulation studies of a highly nonlinear biochemical reactor (Dadhe *et al.*, 2004). The nominal neural network is

$$y_k = 0.43 + 2.02 \tanh(0.5 + p_1 x_k) + 2.11 \tanh(-0.77 + p_2 x_k) + e_k \tag{18}$$

with the parameter vectors $p_1 = \begin{bmatrix}-0.26 & 1.59 & 0.25 & 0.05\end{bmatrix}$, $p_2 = \begin{bmatrix}2.54 & -3.42 & -0.35 & -0.05\end{bmatrix}$ and the input vector $x_k = \left[y_{k-1}\ y_{k-2}\ u_{k-1}\ u_{k-2}\right]^T$ consisting of lagged input and output variables. All initial values y_k and u_k for $k \leq 0$ are set to zero.

4.2 *Signal Specifications*

The input signal for the training data set is an amplitude modulated pseudo random binary signal (APRBS) with a maximum of 1.0, a minimum of 0.5 and a switching time that is equally distributed over the interval $k \in [10; 50]$. For the error process E_k different distributional assumption are made:

(1) Normal distribution $e_k \sim \mathcal{N}(0, 10^{-5})$
(2) Mixed normal distribution $e_{k,1} \sim \mathcal{N}(-2 \cdot 10^{-3}, 10^{-5})$ with probability $p_1 = 0.2$ and $e_{k,2} \sim \mathcal{N}(5 \cdot 10^{-4}, 10^{-5})$ with probability $p_2 = 0.8$.
(3) GARCH(1,1) distribution $e_k = \sqrt{h_k}\nu_k$ with $\nu_k \sim \mathcal{N}(0,1)$ and $h_k = 5 \cdot 10^{-6} + 0.15e_{k-1}^2 + 0.8h_{k-1}$ and initial values $h_0 = e_0 = 5 \cdot 10^{-6}/(1 - 0.15 - 0.8)$, which is the unconditional variance.

4.3 *Neural Network Training*

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

- Neural network training, simulation and validation was performed using the Matlab Neural Network Toolbox. The neural networks consisted of one hidden layer with two neurons and sigmoidal activation function.
- The maximum likelihood estimation of the GARCH parameters was performed with the Matlab SQP solver `fmincon`.
- The nominal data set consists of 200 subsequent data points used for training. Conditional on the state at the end of the training data set, 500 predictions of length 20 with different realizations of the error process E_k were performed. These data were used to evaluate the actual coverage of the bootstrap prediction intervals. The number B of bootstrap resamples and neural network models is 199.

4.4 *Monte Carlo Results*

The results of the Monte Carlo simulation are shown in Tab. 1 for a nominal coverage of 95%. It can be seen, that for a normal distributed error process E_k the *PU-B* method is superior because the *C-B* method neglects the stochastic nature of the neural network parameters which plays an important role here as the training data sets are relatively small.

The *GARCH-B* method seems advantageous when the normality assumption of the error process is violated. Generally, it has better matches in nominal and actual coverage with smaller interval lengths. Nevertheless, the difference between the two methods bootstrapping the parameter estimates is not significant.

In Fig. 1 the quantiles of the bootstrap distribution are depicted as functions of the number of prediction steps. The ability of the bootstrap to account for non-symmetric prediction intervals ensures good coverage with comparatively narrow intervals.

The formulation of the prediction intervals in eq. (4-5) assumes that the upper and lower tails of the distributions have equal quantiles. The results give no clear indication of which method to obtain the empirical distribution of $\hat{y}_{k+j}$ performs best here. In all cases, the Monte Carlo estimate of the standard error is comparatively low. Tab. 2 shows the difference between Efron's and Hall's percentile method. It can be seen that Efron's method performs better than the method based on normalizing the bootstrap distribution. This contradicts other results reported in literature and therefore needs further investigations.

5. CONCLUSION

In this paper, methods have been introduced to evaluate the prediction uncertainty of neural networks 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. 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 anymore, the *GARCH-B* method should perform better than the other bootstrap methods presented here. Further research is necessary to evaluate the proposed methods when (defined) plant-model mismatches are present. The results shown here and in (Dadhe *et al.*, 2004) promise that the bootstrap methods perform properly even with

Method	m	$\bar{\gamma}^{*b}$ [100%]	$\bar{\gamma}^{*b}_{\text{low}}$ [100%]	$\bar{\gamma}^{*b}_{\text{up}}$ [100%]	$\bar{\lambda}$
Normal distribution					
C-B	5	91.08	6.80	2.12	0.0351
PU-B		95.60	3.60	0.80	0.0391
GARCH-B		92.88	6.48	0.64	0.0371
C-B	20	89.92	5.92	4.16	0.0391
PU-B		92.04	4.84	3.12	0.0397
GARCH-B		90.00	7.24	2.76	0.0377
Mixed normal distribution					
C-B	5	94.44	3.84	1.72	0.0355
PU-B		96.88	2.44	0.68	0.0416
GARCH-B		96.24	3.16	0.60	0.0412
C-B	20	97.76	0.48	1.76	0.1570
PU-B		98.24	0.24	1.52	0.1646
GARCH-B		97.84	0.24	1.92	0.1200
GARCH distribution					
C-B	5	91.68	2.80	5.52	0.1325
PU-B		93.20	2.80	4.00	0.1466
GARCH-B		91.80	3.24	4.96	0.1342
C-B	20	91.32	3.04	5.64	0.2157
PU-B		92.48	1.92	5.60	0.2543
GARCH-B		92.00	3.68	4.32	0.2494

Table 1. The table presents the results for nominal coverage of 95%. $100\bar{\gamma}^{*b}\%$ is the Monte Carlo estimate of the actual coverage, $100\bar{\gamma}^{*b}_{\text{low}}\%$ and $100\bar{\gamma}^{*b}_{\text{up}}\%$ are the lower and upper quantiles, respectively, $\bar{\lambda}$ is the estimated prediction interval length from eq. (4).

	80%		90%		95%	
m	Efron	Hall	Efron	Hall	Efron	Hall
1	82.3	78.5	92.0	87.8	96.2	94.0
5	85.8	86.0	93.5	92.6	96.9	96.6
20	90.1	75.9	96.0	84.5	98.2	88.3

Table 2. Comparison of Efron's and Hall's prediction intervals (nominal vs. actual coverage) for the *GARCH-B* method with mixed normal error distribution.

highly nonlinear systems and that the application within neural network based NMPC schemes can improve the controller performance.

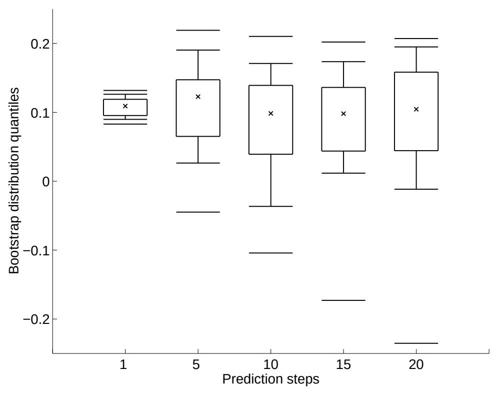

Fig. 1. Bootstrap quantiles using Efron's percentile interval method from eq. (4). The box is the 80% prediction interval and the upper and lower lines represent the 95% and 99% bounds, respectively. The results are shown for the *GARCH-B* method applied to simulations with GARCH error distributions.

6. REFERENCES

Bishop, C. M. (1995). *Neural Networks for Pattern Recognition.* Oxford University Press.

Bollerslev, T. (1986). Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics* **31**, 307–327.

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

Dadhe, K., R. Gesthuisen and S. Engell (2004). Estimating the prediction uncertainty of dynamic neural network process models. In: *7th International Symposium on Dynamics and Control of Process Systems.* Cambridge, MA. Paper No. 148.

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

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

Pascual, L., J. Romo and E. Ruiz (2000). Forecasting returns and volatilities in GARCH processes using the bootstrap. Technical report. Working Paper 00-68(31), Econometrics and Statistics Series, Universidad Carlos III de Madrid.

Pascual, L., J. Romo and E. Ruiz (2001). Effects of parameter estimation on prediction densities: a bootstrap approach. *International Journal of Forecasting* **17**, 83–103.

Rivals, I. and L. Personnaz (2000). Construction of confidence intervals for neural networks based on least squares estimation. *Neural Networks* **13**, 463–484.

Roßmann, V. (2002). Prädiktive Regelung Neuronaler Netze (Predictive Control with Neural Networks). Dr.-Ing Dissertation. Universität Dortmund, Process Control Laboratory, Shaker Verlag, Aachen.

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.