

# Adding Input Noise to Increase the Generalization of Neural Networks is a Bad Idea

Jürgen Van Gorp, Johan Schoukens and Rik Pintelon

ANNIE '98, Intelligent Engineering Systems Through Artificial Neural Networks, Volume 8., pp. 127 - 132.

**Abstract** — In the literature and in commercial software most of the Neural Networks (NN) models are trained using a simple Output Error (OE) cost function. This OE approach may lead to severe bias errors on the predicted output when noisy input data is used. This paper proposes a solution to this problem if input noise cannot be avoided, using the Errors-In-Variables (EIV) approach that is currently used in system identification. In this paper interpolation is suggested as a better alternative when input noise can be avoided.

**Index Terms** — Neural Networks, nonlinear system identification

## I. INTRODUCTION

Consider a system that is measured with true input samples  $u_{0,k} \in \mathfrak{R}$  and true output samples  $y_{0,k} \in \mathfrak{R}$  with  $k = 1 \dots N$  and  $N$  is the total number of measurements. Assume that only a small number of samples could be measured with respect to the number of Neural Network (NN) parameters, such that gaps exist between the different samples and overfitting of the NN is possible. A popular way of artificially increasing the number of measurements is adding jitter, which is the addition of noise on the inputs to improve the generalisation of the network. In the most general case noise is added to both inputs and outputs (Fig. 1).

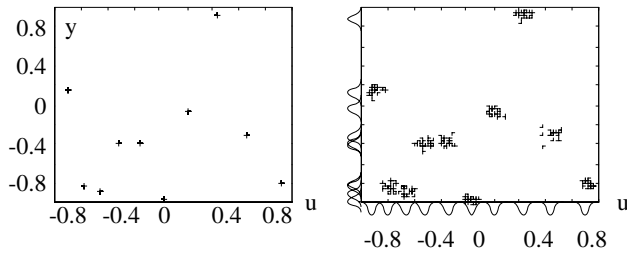


Fig. 1 Increasing the number of measurement using noise

The idea behind jitter is that the weights are shrunk [3]. In section II it will be shown that jitter will actually suppresses higher order derivatives of the NN transfer function, but also that bias is introduced if the Least Squares (LS) cost function is used. This bias can result in significant errors on the NN parameters. In section III. a novel cost function is introduced, based on the Errors-In-Variables (EIV) approach. This cost function was already used to identify linear systems in [4] and used for nonlinear systems in [6]. Examples on the use of the EIV approach will be

given, using a multilayer perceptron NN with a  $\tanh(\bullet)$  hidden layer. Although EIV will help reducing bias effects due to noisy inputs, the best way is to just avoid bias. Section IV gives some examples how extra measurement points can be created using classic interpolation methods. Afterwards it will still be possible to learn the data using a NN in order to have a compact representation of the data.

## II. BIAS EFFECTS DUE TO INPUT NOISE

The Least Squares (LS) costfunction is defined as

$$K_{LS} = \sum_{k=1}^N (y_{0,k} - f_{NN}(u_{0,k}, \theta))^2 \quad (1)$$

with  $\theta$  the parameters of the Neural Network  $f_{NN}$ . Define  $\theta^*$  as the optimal solution of this cost function, such that

$$\theta^* = \underset{\theta}{\operatorname{argmin}} \left( \sum_{k=1}^N (y_{0,k} - f_{NN}(u_{0,k}, \theta))^2 \right) \quad (2)$$

The question is what happens in the most general case where all input and output measurements are taken  $M$  times and noise is added deliberately. Define  $x_k^{[r]}$  and  $y_k^{[r]}$  with  $r = 1 \dots M$  such that

$$u_k^{[r]} = u_{0,k} + \Delta u_k^{[r]}, \quad u_k^{[r]}, \Delta u_k^{[r]} \in \mathfrak{R} \quad (3)$$

$$y_k^{[r]} = y_{0,k} + \Delta y_k^{[r]}, \quad y_k^{[r]}, \Delta y_k^{[r]} \in \mathfrak{R} \quad (4)$$

*Assumption 1*  $\Delta u_k^{[r]}$  and  $\Delta y_k^{[r]}$  are independent, mutually uncorrelated, zero mean Gaussian distributed random variables with known standard deviations  $\sigma_{u,k}$  and  $\sigma_{y,k}$  and  $\sigma_{xy,k} = 0; \forall k$ .

With the added noise samples the new cost function  $K_{LS}$  becomes:

$$K_{LS} = \sum_{k=1}^N \sum_{r=1}^M (y_k^{[r]} - f_{NN}(u_k^{[r]}, \theta))^2 \quad (5)$$

Under assumption 1 this cost function will converge strongly to its expected value

$$E\{K_{LS}\} = E\left\{ \sum_{k=1}^N \sum_{r=1}^M (y_k^{[r]} - f_{NN}(u_k^{[r]}, \theta))^2 \right\} \quad (6)$$

as  $M$  increases tot infinity [2]. Hence to study the influence of the added noise samples for  $M \rightarrow \infty$  it is sufficient to study  $E\{K_{LS}\}$  which is more tractable than (5).

The expected value  $E\{\bullet\}$  is defined as

$$E\{g(x)\} = \int_{-\infty}^{+\infty} g(x)f_d(x)dx \quad (7)$$

with  $f_d(x)$  the probability density function of a stochastic variable  $x \in \mathfrak{R}^m$ . Replace the NN transfer function in (5) by its Taylor expansion

$$\begin{aligned} & f_{NN}(u_{0,k} + \Delta u_k^{[r]}, \theta) \\ &= f_{NN}(u_{0,k}, \theta) + \sum_{i=1}^{\infty} \frac{(\Delta u_k^{[r]})^i \partial^i f_{NN}(u_{0,k}, \theta)}{(i)! (\partial u_{0,k})^i} \end{aligned} \quad (8)$$

Taking into account that  $E\{(\Delta y_k^{[r]})^{2i+1}\} = 0$ ,  $E\{\Delta y_k^{[r]}\} = 0$  and knowing that [5]:

$$\frac{1}{\sigma_u \sqrt{2\pi}} \int_{-\infty}^{+\infty} (\Delta u_k)^{2r} e^{-\frac{(\Delta u_k)^2}{2\sigma_u^2}} d(\Delta u_k) = \frac{\sigma_u^{2r} (2r)!}{2^r r!} \quad (9)$$

the expected value of the cost function (6) becomes:

$$\begin{aligned} E\{K_{LS}\} &= M \sum_{k=1}^N (y_{0,k} - f_{NN}(u_{0,k}, \theta))^2 \\ &+ M \sum_{k=1}^N (\varepsilon_k^2 - 2\mu_k (y_{0,k} - f_{NN}(u_{0,k}, \theta))) \end{aligned} \quad (10)$$

The full derivation of  $\varepsilon_k^2$  and  $\mu_k$  is given in Appendix 1. Here only the second order terms will be considered to simplify the analysis:

$$E\{K_{LS}\} \cong M \sum_{k=1}^N \left[ (y_{0,k} - f_{NN}(u_{0,k}, \theta))^2 \right. \quad (11.a)$$

$$\left. + \sigma_{y,k}^2 + \sigma_{u,k}^2 \left( \frac{\partial f_{NN}(u_{0,k}, \theta)}{\partial u_{0,k}} \right)^2 \right. \quad (11.b)$$

$$\left. - (y_{0,k} - f_{NN}(u_{0,k}, \theta)) \sigma_{u,k}^2 \frac{\partial^2 f_{NN}(u_{0,k}, \theta)}{(\partial u_{0,k})^2} \right] \quad (11.c)$$

The first term (11.a) equals the undistorted cost function (1). (11.b) expresses that the noise terms will cause biasing effects which will be lessened by suppressing the higher order derivatives. These bias effects could be diminished due to the (11.c) term, but if the error terms  $y_{0,k} - f_{NN}(u_{0,k}, \theta)$  can be assumed to have values that are spread around zero and as such have alternating signs, the value of (11.c) will approximately increase with  $\sqrt{N}$ , while (11.b) increases with  $N$ . This means that in most practical cases, adding input noise will suppress higher order derivatives, but also that the true NN parameters  $\theta^*$  will not longer be reached. Both effects will increase with the amplitude of the added noise  $\sigma_{u,k}^2$ . Remark that the output noise does not affect the estimated NN parameters asymptotically ( $M \rightarrow \infty$ ). It adds a  $\theta$ -independent constant to the expected value of the cost function but the point where the cost function reaches its

minimum with respect to  $\theta$  is not changed. Therefore, adding output noise won't cause bias, but it will not help generalization either when applied with the OE cost function.

## Example

To illustrate the theory, a Neural Network was trained on the data points, shown in Fig. 1. The points were created using an arbitrary nonlinear function

$$y_k = \sin(-1 + 7 \sin^5(e^{u_k})) \quad (12)$$

which is sampled at 10 points and fitted using a Single Input Single Output (SISO) NN model with 10 neurons

$$f_{NN}(u_k, \theta) = W_2^T \tanh(W_1^T u_k + B_1) + B_2 \quad (13)$$

in which the  $\tanh(\bullet)$  is taken element wise and

$$\theta = [W_1^T \ B_1^T \ W_2^T \ B_2^T]^T \quad (14)$$

A NN mapping using noiseless data points is shown in Fig. 2 The dotted line is the real function, the solid line is the NN mapping and the crosses depict the simulation samples.

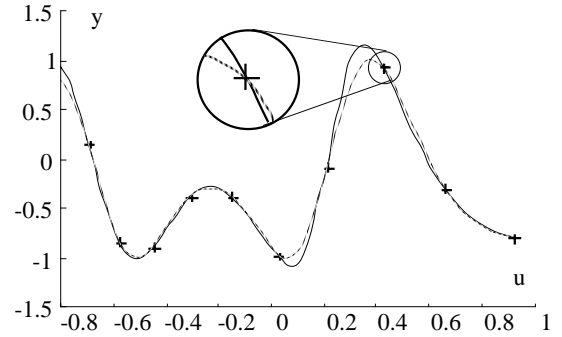


Fig. 2 NN mapping of an arbitrary function with noiseless datapoints

Now consider the case where regularization is considered to avoid overfitting. To fill in the gaps between the measurement points, 40 noise realisations ( $M = 40$ ) are added to each input sample with a standard deviation of  $1/15$ . This corresponds to a signal to noise ratio of 6 dB.

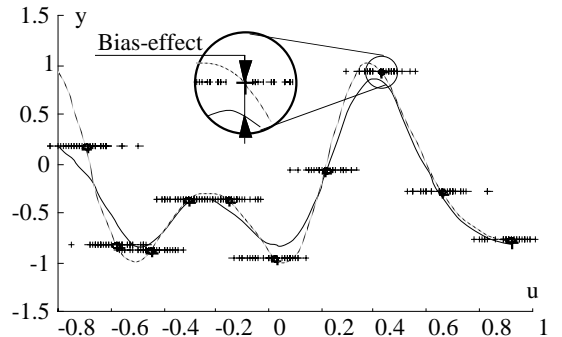


Fig. 3 NN mapping with added noise

The results of this mapping are shown in Fig. 3 The crosses are the new simulation points (only a limited number of samples are shown in the figure).

### Higher order derivatives

The effects of the higher order derivatives with respect to  $u_k$  in equation (11.a) should not be underestimated. In opposition to linear systems, where the second derivative becomes zero, the values of the higher order derivatives in nonlinear systems tend to increase with the order. Rewrite equation (13) as

$$f_{NN}(u_k, \theta) = \sum_{i=1}^m (w_2^{(i)} \tanh(w_1^{(i)} u_k + b_1^{(i)})) + b_2 \quad (15)$$

with  $m$  the number of neurons and  $B_2 = [b_2]$ ,  $W_1 = [w_1^{(1)}, w_1^{(2)}, \dots, w_1^{(m)}]^T$   $B_1 = [b_1^{(1)}, b_1^{(2)}, \dots, b_1^{(m)}]^T$  and  $W_2 = [w_2^{(1)}, w_2^{(2)}, \dots, w_2^{(m)}]^T$  for a SISO system.

It is possible to calculate the  $2j$ -th derivative with respect to  $u_k$  of this NN as

$$\frac{\partial^{2j} f_{NN}(u_k, \theta)}{(\partial u_k)^{2j}} = \sum_{i=1}^m w_2^{(i)} \frac{\partial^{2j} \tanh(a_k^{(i)})}{(\partial a_k^{(i)})^{2j}} (w_1^{(i)})^{2j} \quad (16)$$

with

$$a_k^{(i)} = w_1^{(i)} u_k + b_1^{(i)} \quad (17)$$

The higher order derivatives of the  $\tanh(\cdot)$  part in this equation can become very large. This is shown in Fig. 4 for the first 5 derivatives. Table 1 gives the absolute maximum values for the derivatives up to the eleventh order.

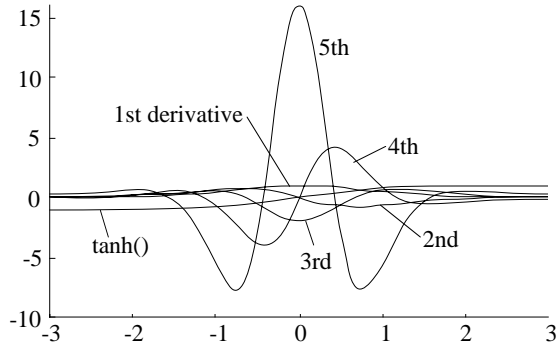


Fig. 4 Higher order derivatives of  $\tanh(\cdot)$

ORDER	MAX VALUE
1	1.000
2	0.7698
3	2.000
4	4.0858
5	16.008
6	52.266
7	272.00
8	1223.7
9	7935.8
10	45569
11	357800

Table 1 Maximum values of the derivatives of  $\tanh(\cdot)$

In [3] is claimed that too little jitter will have little effect. From these examples however can be seen that larger noise levels will cause significant bias effects.

### III. REDUCING BIAS USING ERRORS-IN-VARIABLES

The Errors-In-Variables (EIV) cost function tries to avoid bias in the nonlinear term by using the exact, but unknown input values. The input values are then estimated, together with the NN parameters. The new cost function becomes:

$$K_{EIV} = \sum_{k=1}^N \sum_{r=1}^M \left[ \frac{(y_k^{[r]} - f_{NN}(\hat{u}_{0,k}, \theta))^2}{\sigma_{y,k}^2} + \frac{(\hat{u}_{0,k}^{[r]} - u_k^{[r]})^2}{\sigma_{u,k}^2} \right] \quad (18)$$

The added terms are the variance on inputs and outputs  $\sigma_{u,k}^2$  and  $\sigma_{y,k}^2$ , and the  $\hat{u}_{0,k}$  values which are estimates of the real  $u_{0,k}$  values, such that

$$u_k^{[r]} = \hat{u}_{0,k}^{[r]} + \Delta \hat{u}_{0,k}^{[r]} \quad (19)$$

*Assumption 2* It will be assumed that

$$\hat{u}_{0,k}^{[r]} \cong u_{0,k} \quad (20)$$

Remark that in practise this will not be the case. Each added  $\hat{u}_{0,k}^{[r]}$  introduces an extra parameter, so that in practise biasing effects are still possible when using the EIV cost function. Still, this assumption allows for a good idea about the NN and the bias effects will be less than when using the plain OE cost function.

Remark also that in equation (18) noise on the outputs is considered, such that a normalization is possible on both terms of the cost function. In normal circumstances the EIV cost function is to be used when the inputs are known to contain measurement noise. In this paper EIV will be used to reduce the bias effects.

In [7] the EIV cost function was minimized using a Levenberg-Marquardt (LM) optimization scheme. The drawback of LM learning with EIV is that the presence of the estimated  $\hat{u}_{0,k}$  values in the parameter space will lead to excessive memory needs. This is due to the necessity of pseudo-inverting the very large Jacobian matrix. Further will be shown how optimization is done using backpropagation.

*Lemma 1* Under assumption 2 the  $\arg \min$  with respect to  $\theta$  of the Errors-In-Variables cost function is unbiased when using nonlinear systems.

*Proof:* Following the same reasoning as in (5) calculate the expectation value of the EIV cost function (18)  $E\{K_{EIV}\}$  and replace  $u_k^{[r]}$  and  $y_k^{[r]}$ . This gives that

$$E\{K_{EIV}\} = \sum_{k=1}^N \sum_{r=1}^M \left[ \frac{(y_{0,k} + \Delta y_k^{[r]} - f_{NN}(\hat{u}_{0,k}^{[r]}, \theta))^2}{\sigma_{y,k}^2} + \frac{(\hat{u}_{0,k}^{[r]} - \hat{u}_{0,k}^{[r]} - \Delta \hat{u}_{0,k}^{[r]})^2}{\sigma_{u,k}^2} \right] \quad (21)$$

such that

$$E\{K_{EIV}\} = M \sum_{k=1}^N \frac{(y_{0,k} - f_{NN}(\hat{u}_{0,k}^{[r]}, \theta))^2}{\sigma_{y,k}^2} + M \sum_{k=1}^N \frac{\sigma_{y,k}^2}{\sigma_{y,k}^2} + M \sum_{k=1}^N \frac{\sigma_{u,k}^2}{\sigma_{u,k}^2} \quad (22)$$

and finally, with Assumption 2

$$E\{K_{EIV}\} \cong M \sum_{k=1}^N \frac{(y_{0,k} - f_{NN}(u_{0,k}, \theta))^2}{\sigma_{y,k}^2} + 2NM \quad (23)$$

The expectation of the EIV cost function with noisy inputs is the same as the Bayesian form of the LS cost function and they will have the same minimum with respect to  $\theta$ .  $\square$

To obtain the optimization step (in NN this is called the learning rule), here the backpropagation gradient method is considered with

$$\Delta \theta_{EIV} = -\eta J^T E \quad (24)$$

in which the parameter vector  $\theta_{EIV}$  now also contains the estimated input samples, or

$$\theta_{EIV} = [W_1^T \ B_1^T \ W_2^T \ B_2^T \ \hat{u}_0^T]^T \quad (25)$$

The error vector  $E$  also contains the errors on the estimated values for  $\hat{u}_0$ , or

$$E = \begin{bmatrix} \frac{y_k^{[r]} - f_{NN}(\hat{u}_{0,k}^{[r]}, \theta)}{\sigma_{y,k}} \\ \frac{\hat{u}_{0,k}^{[r]} - u_k^{[r]}}{\sigma_{u,k}} \end{bmatrix} = \begin{bmatrix} E_{f_{NN}} \\ E_u \end{bmatrix} \quad (26)$$

The error vector has as size  $(NM + m) \times 1$  which is already much larger than the original  $m \times 1$  vector that was used in the OE case. If a SISO system with 10 neurons would be trained with 1000 measurement samples, this would mean that the error vector grows from 31 elements to 1031 elements. The difference becomes even more apparent when the Jacobian  $J$  is considered. In [8] it is shown that for EIV the Jacobian will become

$$J = \begin{bmatrix} \frac{\partial f_{NN}}{\partial \theta} & \frac{\partial f_{NN}}{\partial \hat{u}_0} \\ 0 & I \end{bmatrix} = \begin{bmatrix} A & B \\ 0 & I \end{bmatrix} \quad (27)$$

In the OE case  $J$  would only consist of the  $A$  matrix which has a size of 31000 elements in the example above. In the case of EIV the Jacobian will have over two million elements. It is however possible to make use of the property that the Jacobian has many zeros in it. Putting equations (27) and (26) in equation (24) gives the following learning rule

$$\Delta \theta_{EIV} = -\eta \begin{bmatrix} A^T E_{f_{NN}} \\ B^T E_{f_{NN}} + E_u \end{bmatrix} = -\eta \begin{bmatrix} \left(\frac{\partial f_{NN}}{\partial \theta}\right)^T E_{f_{NN}} \\ \left(\frac{\partial f_{NN}}{\partial \hat{u}_0}\right)^T E_{f_{NN}} + E_u \end{bmatrix} \quad (28)$$

$B^T E_{f_{NN}}$  can be calculated without the explicit knowledge of the large  $B$  matrix. The result is that the memory requirements are significantly reduced for backpropagation and other gradient methods. To increase learning speed and robustness, techniques like an adaptive learning rate  $\eta$  and momentum term are applicable. In practise the EIV costfunction shows many local minima and convergence is not always guaranteed. Further a learning scheme is given that will use the EIV method as a postprocessing tool after using the OE method.

EIV allows to shape both input and output measurements. Observations showed that the EIV method can be prone to overfitting. Normally overfitting is prevented by creating a second set of measurements (called the validation set) and stopping the learning whenever the cost function applied on the validation set starts to raise again. The method applies also for EIV, but in [8] it is shown that instead of the LS cost function the following cost function should be used:

$$C_{EIV} = \sum_{l=1}^{N_v} \frac{(y_l - f_{NN}(u_l, \theta))^2}{\sigma_{y,l}^2 + \left(\frac{\partial f_{NN}(u_l, \theta)}{\partial \hat{u}_0}\right)^2 \sigma_{u,l}^2} \quad (29)$$

with  $N_v$  the number of measurements in the validation set. Remark that in this cost function the measured input and output values are used, instead of the estimated values. This approach makes the validation test fast enough for practical use. Remark that the partial derivative used in the denominator is of the same form as the derivative used to calculate the  $B$  matrix in equation (28), so that the same routines can be applied.

A suggested optimization scheme is the following:

- Use the OE method with random initial NN parameter values to obtain the OE  $\hat{\theta}$  vector, with

$$\hat{\theta} = \underset{\theta}{\operatorname{argmin}} \left( \sum_{k=1}^N (y_k - f_{NN}(u_k, \theta))^2 \right) \quad (30)$$

- Calculate the initial parameters for the EIV method as  $\theta_{EIV} = [\hat{\theta}^T \ u_l^T]^T$  and start

optimizing using equation (28).

After each step calculate the cost function (29) and stop the learning process whenever this cost function starts to increase.

### Illustrative example

To demonstrate the use of EIV, the same example as in section II is taken with the same NN model. The initial parameters for EIV are taken as given in the above optimization scheme. The  $A$  matrix is built from four matrices containing the partial derivatives for  $W_1$ ,  $B_1$ ,  $W_2$  and  $B_2$

$$A = \begin{bmatrix} \frac{\partial f_{NN}}{\partial W_1} & \frac{\partial f_{NN}}{\partial B_1} & \frac{\partial f_{NN}}{\partial W_2} & \frac{\partial f_{NN}}{\partial B_2} \end{bmatrix} \quad (31)$$

with

$$\frac{\partial f_{NN}}{\partial W_1} = \begin{bmatrix} \frac{\partial f_{NN}(u_1, \theta)}{\partial w_1^{(1)}} & \frac{\partial f_{NN}(u_1, \theta)}{\partial w_1^{(2)}} & \dots \\ \frac{\partial f_{NN}(u_2, \theta)}{\partial w_1^{(1)}} & \frac{\partial f_{NN}(u_2, \theta)}{\partial w_1^{(2)}} & \dots \\ \dots & \dots & \dots \end{bmatrix} \quad (32)$$

and

$$\frac{\partial f_{NN}(u_k, \theta)}{\partial w_1^{(i)}} = w_2^{(i)} \frac{\partial \tanh(a_k^{(i)})}{\partial a_k^{(i)}} u_k \quad (33)$$

In an analogous way the other terms are found as

$$\frac{\partial f_{NN}(u_k, \theta)}{\partial b_1^{(i)}} = w_2^{(i)} \frac{\partial \tanh(a_k^{(i)})}{\partial a_k^{(i)}} \quad (34)$$

$$\frac{\partial f_{NN}(u_k, \theta)}{\partial w_2^{(i)}} = a_k^{(i)} \quad (35)$$

$$\frac{\partial f_{NN}(u_k, \theta)}{\partial b_2^{(i)}} = 1 \quad (36)$$

The  $B$  matrix is diagonal and contains

$$\frac{\partial f_{NN}}{\partial W_1} = \begin{bmatrix} \frac{\partial f_{NN}(u_1, \theta)}{\partial u_1} & 0 & \dots \\ 0 & \frac{\partial f_{NN}(u_2, \theta)}{\partial u_1} & \dots \\ \dots & \dots & \dots \end{bmatrix} \quad (37)$$

such that

$$\frac{\partial f_{NN}(u_k, \theta)}{\partial u_i} = w_2^{(i)} \frac{\partial \tanh(a_k^{(i)})}{\partial a_k^{(i)}} w_1^{(i)} \delta_{ik} \quad (38)$$

with  $\delta_{ik}$  the Kronecker symbol. The results of this

mapping can be seen in Fig. 5. Only a limited number of simulation samples are shown in this figure for clarity. Remark that noise is added to both inputs and outputs so that the learning algorithm is able to move the samples more freely. The dotted line is the true function. As can be seen in the figure, the overall performance of the EIV estimate is better.

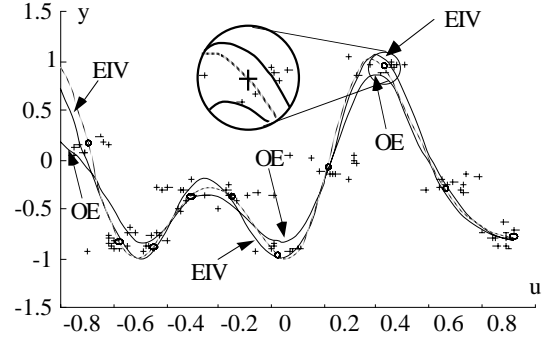


Fig. 5 EIV mapping compared to OE mapping

## IV. GENERALIZATION USING INTERPOLATION

The samples that are used for learning in Fig. 3 have much resemblance to a zero order interpolation of the measured samples. One might wonder why it is needed to go through the pain of choosing the amplitude of noise that is needed to obtain good generalization, if interpolation is available. In Neural Networks  $\sigma_u^2$  is called the ridge value and both statistical as empirical methods exist to obtain this ridge value [3]. Choosing the ridge value too high will cause overlapping  $u_k^{[r]}$  regions with an increasing danger of bias effects. Putting jitter on measurement points makes that the experimenter is less aware of what will happen with the NN mapping. It is a non-proven belief that adding noise on the loose will make things better.

The question remains how a good mapping can be achieved if the input samples are known to be exact (noiseless), but the number of samples are too few to guarantee a good NN mapping in between the given measurements. The best solution would be to perform additional measurements. If that is not possible, it is the experimenter who should be able to give additional information on the system. We suggest that this additional information is given explicitly by means of the order of a polynomial approximation to create intermediate points prior to mapping the NN. The Neural Network is then used as a compressing tool that is used to avoid storing long interpolation tables. Examples of this are shown in the following figures for a zero to third order interpolation between the data points. The dotted lines represent the true functions, while the solid line is the NN approximation. Remark that the NN tries to follow the interpolated points

and that the interpolation errors can be fully explained by the lack of measurement points. Remark also that the NN does not exactly meet the measured values. In this case however, this is only caused by a nonperfect mapping and it can be overcome by choosing a higher order (smoother) interpolation method.

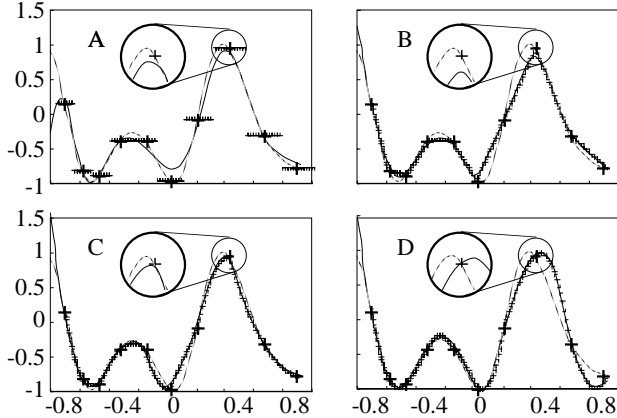


Fig. 6 NN mapping with interpolated points. A) zero order B) first order C) second order D) third order interpolation

## V. CONCLUSIONS

In this paper it has been shown that adding noise to measurements to improve generalisation can cause severe uncontrolled bias errors on the predicted output with the mostly used Output Error cost function. To reduce this bias the Errors-In-Variables cost function is introduced and its application is described for generalization purposes using an example. This paper stated that even when it is possible to reduce the bias, adding noise should be avoided. To increase generalization this paper suggests to use interpolation techniques where the experimenter chooses the order of an interpolation polynomial. Examples are given to demonstrate this technique.

**Appendix 1** Expected value of the OE costfunction (5).

Put the Taylor expansion (8) in equation (6). This gives:

$$E\{K_{LS}\} = E\left\{\sum_{k=1}^N \sum_{r=1}^M \left[ y_{0,k} + \Delta y_k^{[r]} - f_{NN}(u_{0,k}, \theta) - \sum_{i=1}^{\infty} \frac{(\Delta u_k^{[r]})^i}{(i)!} \frac{\partial^i f_{NN}(u_{0,k}, \theta)}{(\partial u_{0,k})^i} \right]^2\right\} \quad (39)$$

After computing the summation terms and knowing that

$$E\{(\Delta u_k^{[r]})^{2i+1}\} = 0 \quad (40)$$

$$E\{\Delta y_k^{[r]}\} = 0 \quad (41)$$

$$E\{(\Delta y_k^{[r]})^2\} = \sigma_{y,k}^2 \quad (42)$$

equation (10) is derived:

$$E\{K_{LS}\} = M \sum_{k=1}^N (y_{0,k} - f_{NN}(u_{0,k}, \theta))^2 + M \sum_{k=1}^N \left[ \varepsilon_k^2 - 2\mu_k (y_{0,k} - f_{NN}(u_{0,k}, \theta)) \right] \quad (43)$$

with

$$\varepsilon_k^2 = \sigma_{y,k}^2 + \sum_{i=1}^{\infty} \sigma_{u,k}^{2i} \frac{(2i)!}{2^i (i!)^3} \left( \frac{\partial^i f_{NN}(u_{0,k}, \theta)}{(\partial u_{0,k})^i} \right)^2 + 2 \sum_{i=2}^{\infty} \sum_{j=1}^{i-1} \frac{\sigma_{u,k}^{2i} (2i)!}{(i!)^2 2^i (2j-i)!} \frac{\partial^j f_{NN}(u_{0,k}, \theta)}{(\partial u_{0,k})^j} \frac{\partial^{2i-j} f_{NN}(u_{0,k}, \theta)}{(\partial u_{0,k})^{2i-j}} \quad (44)$$

and

$$\mu_k = \sum_{i=1}^{\infty} \frac{\sigma_{u,k}^{2i}}{2^i (i)!} \frac{\partial^{2i} f_{NN}(u_{0,k}, \theta)}{(\partial u_{0,k})^{2i}} \quad (45)$$

□

## Acknowledgments

This paper presents research results of the Belgian Programme on Interuniversity Poles of Attraction (IUAP 4/2), initiated by the Belgian State, Prime Minister's Office for Science, Technology and Culture; and the Flemish Government (GOA, IMMI2). The scientific responsibility rests with its authors.

## References

- [1] Geman S. Bienenstocke E., Doursat R., "Neural Networks and the Bias/Variance Dilemma", *Neural Computing*, 4, pp 1-58, 1992.
- [2] Lukacs, "Stochastic Convergence", Academic Press, New York, 1975
- [3] Sarle Warren, Neural-Nets FAQ, comp.ai.neural-nets, april 1997
- [4] Johan Schoukens and Rik Pintelon, "Identification of Linear Systems", Pergamon Press - London, 1991
- [5] Alan Stuart & J. Keith Ord, "Kendall's Advanced Theory of Statistics, Distribution Theory", Vol. 1, Charles Griffin & Co., 1987
- [6] Gerd Vandersteen, "Identification of Linear and Nonlinear Systems in an Errors-in-Variables Least Squares and Total Least Squares Framework", PhD thesis, April 1997, Vrije Universiteit Brussel, 1050 Brussel, Belgium
- [7] Jürgen Van Gorp, "Control of a Static Nonlinear Plant Using a Neural Network Linearization", IEEE World Congress on Computational Intelligence, IJCNN '98, Anchorage, Alaska, 1998
- [8] Jürgen Van Gorp, "Learning Neural Networks With Noisy Inputs Using The Errors-In-Variables approach", internal note, dept. ELEC, Vrije Universiteit Brussel, 1998