Quadratic Neural Unit and its Network in Validation of Process Data of Steam Turbine Loop and Energetic Boiler

Ivo Bukovsky¹, Martin Lepold,¹,² and Jiri Bila¹

¹ Czech Technical University in Prague, ² I & C ENERGO a.s.

Abstract— This paper discusses results and advantages of the application of quadratic neural units and novel quadratic neural network to modeling of real data for purposes of validation of measured data in energetic processes. A feed forward network of quadratic neural units (a class of higher order neural network) with sequential learning is presented. This quadratic network with this learning technique reduces computational time for models with large number of inputs, sustains optimization convexity of a quadratic model, and also displays sufficient nonlinear approximation capability for the real processes. A comparison of performances of the quadratic neural units, quadratic neural networks, and the use of common multilayer feed forward neural networks all trained by Levenberg-Marquard algorithm is discussed.

I. INTRODUCTION

Control of the power plant performance is performed gradually by the plant operators by using technical indicators like the efficiency, balance of temperatures and emission concentrations etc. [1]. These indicators are not directly measurable and must be calculated from measured values that have usually limited precision and reliability. To achieve real profit by using these technical indicators, it is necessary to use measured data with higher reliability and precision for the power plant control [2]. In many cases, the measured values do not represent the values that should enter into calculation (e.g. incorrect location of the sensor in piping or in a combustion chamber). Raw process data are frequently corrupted by random and gross errors. The data validation changes this raw data to reliable and more precise data free of gross and random errors [3]. Popular methods that are today widely investigated and used for validation (precisiation) of measured data basically originate from concepts of neural network ensembles [4] and bootstrapping [5] [6] and, e.g., [7]. Fundamentally, these concepts are based on redundant modeling of process variables by neural networks. In our paper, we present some results from development of data reconciliation system [8] [9] and we focus in this paper on a class of higher order neural networks (HONN) (or a class of polynomial neural network–PNN) for redundant modeling of process variables. To demonstrate the popularity of HONN or PNN today, we can briefly refer to works [10]-[16]. In this paper, we introduce an application of a special feed forward Quadratic Neural Network (QNN) with individual training of neurons in the first layer and consequent individual training of the output neuron. Quadratic polynomial models can be viewed as midpoints between linear models, which are still popular in technical applications for their solvable and convex (single-solution) nature, and between highly nonlinear models such as, for example, multi-layer perceptron networks that are powerful approximators, but whose approximation strength sometimes, paradoxically, is their drawback for real applications, see the general sketch in Fig. 1. In other words, linear models can not learn nonlinear system while too strong approximating neural networks can unwillingly learn also erroneous data that may worsen the overfitting problem, worsen the local minima issue, etc.. This becomes an issue because the free-of-errors data are usually hardly available by measurement in energetic processes (that is also why the data validation is needed). Ideally, for purposes of process data validation, a cognitive process model should learn from correct data while the model should ideally reject erroneous (random error) data. This might be naturally achieved by sufficient nonlinear approximation capability of a neural architecture to learn major process dependencies, while the network shall be unable to learn unusual and more complicated dependencies introduced by erroneous measurements, see the sketch in Fig. 1. In case of energetic processes, the need for nonlinearity can be assumed lower when compared to ECG or EEG classification or similar problems. Assuming that reasonably correct data are prevailing in the training data set, we introduce the QNN (Fig. 3) and its sequential training of neurons because this together sustain convexity of optimization of quadratic model and also show sufficient (but presumably somewhat limited) nonlinear approximation capability (if we thought of some other neural architectures). In the next section, we recall a building element of QNN that is the static quadratic neural unit (QNU) [17]–[22] and introduce QNN with sequentially trained neurons for modeling of real energetic processes. Then we present and compare results on redundant modeling of the real process data by the QNU, conventional feed forward neural networks (multi layer perceptron – MLP), and with the proposed quadratic network with its various configurations. In the end, we discuss aspects of the use of QNU and its networks (QNN).
from the point of observed approximation.

![Diagram](image)

**Fig. 1:** A principal sketch when too strong neural models (\(\cdots\)) can fit also erroneous data while useful models (\(\approx\)) should not be able to learn erroneous data and should tend to learn only prevailing correct data during optimization to not-too-complicated system (e.g. when quadratic approximation can be sufficient).

II. QUADRATIC NEURAL NETWORK

In this section we propose a novel class of higher order neural networks that benefits from a specific way of two-phase training. To describe the used network we use the notation originally introduced for quadratic neural units in [17] and consequently in [18]–[22]. The building elements of the proposed network are static quadratic neural units and its \(\eta\)th instance is given in Eq.(1) and (2).

\[
x_\eta = \sum_{i=0}^{n} \sum_{j=0}^{n} w_{ij} x_i x_j = x_a^T W_\eta x_a,
\]

where \(x\) denotes \(n \times 1\) vector of inputs into the neuron, \(T\) stands for transposition, \(x_a = [1 \ x]^T\) denotes \((1+n) \times 1\) vector of the inputs augmented with \(x_0 = 1\) that allows neural bias \(w_{0,0}\) and linear terms \(w_{0,1} x_1 + \cdots + w_{0,n} x_n\), and \(W_\eta\) is \((1+n) \times (1+n)\) upper triangular weight matrix shown in Eq.(2) (nonzero elements bellow the main diagonal would introduce merely redundant terms into the polynomial terms of QNU).

\[
x_a = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, \quad W_\eta = \begin{bmatrix} w_{0,0} & w_{0,1} & \cdots & w_{0,n} \\ 0 & w_{1,1} & \cdots & w_{1,n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & w_{n,n} \end{bmatrix}
\]

(2)

As discussed later in this paper, the advantage of the QNU given in Eq.(1) and Eq.(2) is its convexity and nonlinear function approximation capability. Its major disadvantage is the quadratic growth of the number of neural weights with the number of neural inputs (Fig. 2) that can results in the unbearable computational overload (hours and longer) for reasonably priced hardware. In case of energetic processes (turbine loops, large boilers), there are usually more than 20 process variables (model inputs) measured in a real time. It should be mentioned that we consider process data validation based on redundantly modeled data by a number of neural models. Then in case of energetic processes we easily arrive to total of hundreds or even thousands of neural models for all process variables.

![Graph](image)

**Fig. 2:** The quadratic growth of the number of weights of QNU as a function of the number of inputs (Eq.(1), Eq.(2)).

To decrease the computational burden and to benefit from capabilities of neural architectures, we cognitively transform the original \(n \times 1\) input vector \(x\) (of all \(n\) measured variables) as follows. Every original input vector \(x\) with \(n\) elements can be viewed as a composition of shorter sub-vectors \(\chi_1, \chi_2, \ldots, \chi_\zeta\) and a simple example of such decomposition is shown in Eq.(3).

\[
x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} x_{1,1} \\ x_{1,1} \\ \vdots \\ x_{1,n} \end{bmatrix} = \begin{bmatrix} \chi_1 \\ \chi_2 \\ \cdots \\ \chi_\zeta \end{bmatrix}, \quad m < n
\]

(3)

For example, imagine a process with input vector \(x\) of \(n=30\) variables being decomposed into \(\zeta=6\) sub-vectors. In that case, all sub vectors \(\chi_1 \ldots \chi_6\) can have the same number of \(m=5\) elements, see Eq.(3). Then the total number of neural weights of the quadratic neural network (Fig. 3), including weights for bias and linear terms, would be \((1+5)+(1+5)^2)/2+(7+7^2)/2=154\) weights. On the other hand, if we use a single QNU with 30 inputs, the total of needed neural weights would be \((31+31^2)/2=496\) while a conventional feed forward neural network with, e.g., 5 neurons in the hidden layer and a single output neuron (each with linear biased synaptic operation) would need \(31*5+5+1=161\) neural weights (see Fig. 2, Fig. 4).

The network (Fig. 3) is a feed forward neural network with \(\zeta\) quadratic neurons (nodes) in the first layer and one output quadratic neuron; the network training consists of two specific phases. In the first training phase, every \(i^{th}\) neuron in the first layer (Fig. 3) is fed with one augmented sub-vector \(\chi_i\), and trained to the modeled variable separately. That is, after drawing \(\zeta\) sub-vectors out of the vector of input variables \(x\) (Eq.(3)), we augment each sub-vector \(\chi_i\) with unit, thus creating \(\chi_{ia}\), and feed it to its own \(i^{th}\) QNU in
the first layer (Fig. 3). These QNUs are then individually trained.
to the target variable. Thus, the $i^{th}$ QNU in the first layer learns best nonlinear correlation of the $i^{th}$ sub-vector $\chi_i$ with the modeled variable (i.e., each neuron in the first layer is first trained to the target value independently). In the second training phase, only the output neuron (Fig. 3) is trained to the target variable and the already-trained first layer shrinks the original $n \times 1$ input vector $x$ into the shorter $\zeta \times 1$ augmented vector $v$, that is reasonably short ($\zeta < 20$) for fast training of output QNU.

In the proposed design of QNN shown in Fig. 3, we consider the same number of $m$ inputs feeding first layer neurons $QNU_1, \ldots, QNU_{\zeta-1}$ while its last (the bottom) neuron $QNU_{\zeta}$ is fed by the remaining variables from $x$ (whose number can be $\leq m$). Fig. 4 shows the total number of weights of QNN as a function of the number of inputs into first layer neurons and for various lengths of $x$ ($n$ … the number of model-feeding variables). Fig. 4 shows that significantly smaller number of neural weights can be achieved when $m \in [3, 8]$. Theoretically, we may expect the QNU to be a stronger approximator than QNN because neural weights of QNU do not influence (multiply or sum) each other contrary to QNN where this happens via the output neuron.

### III. Experimental Analysis

In this section we show results on redundant modeling of process variables by the quadratic network introduced in section II. For comparison we also show results obtained by conventional feed forward neural networks with one and two hidden layers and we also show results by single QNU trained in a regular way (a whole network at once).

Because of the static nature of the modeled processes (due to relatively long intervals of the available sampling) and partially because of the specific locally stationary nature of the process, i.e., the process can run in the same conditions for a long time in a row, the global optimization of the network was used, namely the Levenberg-Marquardt (LM) algorithm to train all neural architectures. During training, we decreased the learning rate when a training epoch had not resulted in decrease of the performance index (the sum of square errors over an epoch). All data were normalized by subtracting the mean value and by dividing two standard deviations of the training data set before processing by neural networks.

First we show results on redundant modeling of real data measured on a hot steam turbine loop. At every sampling moment, there are available 18 measurements of process variables such as steam pressure, temperature, flow, and actual turbine power. Using static neural network models (QNU, QNN, MLP), each of the 18 variables is individually and redundantly modeled and always the remaining 17 variables feeds the neural models as inputs. It should be mentioned that some of the variables, such as some flows, were obtained indirectly, i.e., calculated by other methods by data supplier [1]–[3] in this steam turbine loop case. Proper explanation of the indirect measurement would exceed the scope of this paper, and it does not relate to the use of neural networks anyway. A typical result is in Fig. 5 showing redundant modeling of the steam flow value (dark line) at every sampling point $k$ (on vertical axis); the process variable was at every sampling point modeled with:

- 10 two-hidden-layer feed forward perceptron NN (neuron
Fig. 5: Redundant modeling of hot steam flow by instances of single QNU and common feed forward neural networks (MLP) (testing trained networks on a new data); the solid line is the measured variable (steam flow) running in time (via horizontal axes) and the darkest points are redundantly modeled values by QNU; the lighter points are redundant values by MLP networks and it shows that MLP have much larger variance of outputs than outputs of QNU.

Fig. 6: Training performances of individually trained first layer neurons consequently followed by training of the output neuron (left) and testing of QNN again on training data (right, 5xQNN) for the same variable as in Fig. 5, the picture demonstrates that instances of trained QNN have very similar outputs.

Fig. 7: Testing redundant models of the hot steam flow in a turbine loop on a new (testing) data by a single QNU (left) and by QNN with sequential learning (right) (for the same variable as in Fig. 5 and Fig. 6), see Eq.3 for $m$; the picture demonstrates that outputs of various trained QNU and QNN are similar.
configurations from 3-2-1 up to 7-3-1, longest training, largest error variance of the models, yellow (lightest) points in the picture),

- 10 single-hidden-layer NN (neuron configurations from 3-1 up to 7-1 networks).... shorter training, smaller error variance (cyan (middle dark) points in the picture),
- 10 single static QNUs (fastest training, smallest error variance of the models, magenta (darkest) points in the picture).

The training on turbine data was made with \( k = 1 \ldots 1881 \) training pairs \([x(k), y_{\text{out}}(k)]\) and next 1000 samples are for testing. In Fig. 5 we can clearly see that in average all QNUs (darkest points) keep closest to the modeled variable (the blue (dark) curve) and we relate this feature to good approximation capability of a single QNU and its convex (quadratic) nature. In the detail in Fig. 5, we can see that the neural models indicate exceptionally lower error than the measured one, which is what the neural networks learned from the training data and this might indicate exceptionally long interval of significant errors of highly imprecise measurement (if the training data set was well selected). Further processing of these neural outputs for data validation [8] [9] exceeds the scope of our paper.

Fig. 6 shows performance of QNNs trained with sequential learning; the left hand side picture shows that first layer neurons were trained with various accuracy and that the finally trained output neuron (being fed by already trained first layer neurons) achieves better performance (the sum of square errors--SSE) than any of the first layer neuron; outputs of 5 QNNs trained from different initial conditions are superimposed on the target variable in the right hand side of Fig. 6. Fig. 7 shows more detailed comparison of the same variable as in Fig. 5 and Fig. 6 redundantly modeled by standalone QNUs (Eq.(1)) and by QNNs (Fig. 3). In the left part a) of Fig. 7, standalone QNUs show better agreement with new measured data except around samples 540–600. Various configurations of QNNs (Fig. 7 b)) indicate slightly worse agreement with values at first samples (e.g. details for samples 38–44) but they show, contrary to QNUs, better agreement with measured samples ~540–600. Recall, that either standalone QNUs or various configurations of QNNs displayed much smaller variance of neural output when tested on new data then MLP networks with linear synaptic operation, sigmoid output function, with either one or two hidden layers (Fig. 5). In appendix, there are examples of redundantly modeled process variables of an energetic boilers by QNNs with various configurations (\( m \in <3,8>\)) and trained from different initial conditions.

IV. DISCUSSION

There are at least two major points that can be open for a discussion. First, the quadratic polynomial models can be interesting for many technical applications due to their capability to approximate nonlinear behavior with certain degree of accuracy while their optimization (learning) has still convex-like nature. This idea is sketched in Fig. 8, where the trained linear model does not reach the accuracy of the quadratic model, and the quadratic model can not be trained as accurately as a more nonlinear neural network can.

Further, a quadratic model can have naturally steeper slope of the “error surface” for optimization, while the more capable nonlinear network will introduce more significant local minima disposed in the weight space (represented by the horizontal axis). Our results with Levenberg-Marquardt (L-M) algorithm confirm these statements; Fig. 5 shows that QNU trained from various initial conditions has smallest variance of outputs (for every data sample) when tested on new data, while MLP networks with one or two hidden layers (of neurons with linear activation functions and sigmoid output function) have much larger variance of the outputs (and also the optimization of QNU was in general significantly faster than MLP as for Fig. 5). This can be supported by results shown in Fig. 6, Fig. 7, Fig. 10, and Fig. 11, where we can see that all redundant quadratic models produce similar outputs with the sufficient degree of accuracy that is interesting for further processing for validation and precision of real time measurements. Those results confirm that QNU and QNN with sequential learning are much more free from local minima issues than MLP networks (the spread of outputs in Fig. 5). Second interesting point for discussion is the approximation strength of various configurations of QNN (various \( m \) – see Eq.(3).) and their comparison to single QNU. Comparison of achieved accuracy on training data after the last training epoch is shown in Fig. 9 (boiler left, turbine right). The standalone QNU is represented by the last value on the horizontal axes (QNU is at \( m = 23 \) for boiler and at \( m = 17 \) for turbine, Fig. 9). Looking at Fig. 9 and comparing it to Fig. 4, we can see that QNNs had lower approximation ability around \( m \in <3,8>\), that is the interval where QNN has also lowest number of neural weights.
Fig. 8: An idealized sketch that represents learning capabilities of a linear model, of QNU or QNN, and of more nonlinear models (e.g., MLP).

Fig. 9: Approximation capability of QNN and QNU plotted as training accuracy of QNN for various \( m \) (see Eq.(3) for \( m \)); QNN becomes QNU at \( m=23 \) for boiler (left) and at \( m=17 \) for turbine (right), (500 training samples, Levenberg-Marquardt, 300 epochs)

We can see also in Fig. 9 that the standalone QNU (left at \( m=23 \), right at \( m=17 \)) was trained most accurately. Nevertheless, the most accurate training does not necessarily mean the best accuracy for testing as show in Fig. 11, and the accuracy of tested QNNs in the region \( m \in [3,8] \) is still very attractive for data validation, see Fig. 10.

V. SUMMARY

The quadratic neural units and networks are the important midpoint between linear (convex) models and common neural networks (too nonlinear NN may suffer from overfitting and local minima issues that may still impede popularity of NN among industrial community). The approximation strength and convex nature of QNU and QNN with sequential learning is attractive for modeling and thus validation of measured data in energetic processes as presented and discussed with data from the turbine loop and the large energetic boiler.

REFERENCES

Fig. 10: Redundant modeling of measured flow of an energetic boiler by eighteen QNNs each sequentially trained with sequential learning; other measured variables were used as inputs to the models (see Fig. 12 for training data).

APPENDIX

Fig. 10: Redundant modeling of measured flow of an energetic boiler by eighteen QNNs each sequentially trained with sequential learning; other measured variables were used as inputs to the models (see Fig. 12 for training data).
Fig. 11: A comparison of single QNU, QNN with $m=5$, and QNN with $m=11$ on modeling of another variable (temperature) of boiler; QNU(dash) is closer to measurement in detail A, while both QNNs are closer to measured values in detail B; (temperature – another variable that is difficult to measure accurately in energetic boilers), (see Fig. 12 for training data).

Fig. 12: The used training data (for the boiler before normalization) for QNNs in Fig. 10 and Fig. 11; total of 24 measured variables were used (always one as a target and remaining 23 as neural model inputs).