Neural Comput & Applic (1999)8:95–105

Neural Computing & Applications

Using Growing RBF-Nets in Rubber Industry Process Control

U. Pietruschka and R.W. Brause

FB Informatik, J.W. Goethe-University, Frankfurt a.M., Germany

This paper describes the use of a Radial Basis Function (RBF) neural network in the approximation of process parameters for the extrusion of a rubber profile in tyre production. After introducing the rubber industry problem, the RBF network model and the RBF net learning algorithm are developed, which uses a growing number of RBF units to compensate the approximation error up to the desired error limit. Its performance is shown for simple analytic examples. Then the paper describes the modelling of the industrial problem. Simulations show good results, even when using only a few training samples. The paper is concluded by a discussion of possible systematic error influences, improvements and potential generalisation benefits.

Keywords: Adaptive process control; Parameter estimation; RBF-nets; Rubber extrusion

1. Introduction

Process control in rubber industry has an image of a 'dirty' branch of industry. This is not only because of the dull and dusty rubber and tyre production rooms where the products are 'baked' by heat and steam, but also because the macromolecular proportions of rubber are hard to predict due to their nonlinear nature. When the rubber mixture leaves an extruder (the melting and form-giving machine) after being heated up to 110–140°C, compressed with 70–140 bar by a screw conveyor and pressed through a metal mask, the rubber relaxes, (i.e. it expands or shrinks), depending on the mixture, thus changing its shape in a non-linear manner by 10–20% up to 50%.

The basic production layout for our tyre profile production example is shown in Fig. 1.

The task of process control consists of estimating the extrusion parameters (i.e. the shape of the extrusion metal mask) necessary for an acceptable rubber product after relaxation. To date, owing to the nonlinear nature of the macromolecular mixture, this task has not been solved analytically. Instead, specialised people estimate the profile of the original metal mask using their subject experience, and correct their estimates as the process develops. This gives a trial-and-error turn-around production cycle.

This kind of production has severe disadvantages for the production business:

- The start for a new product is delayed by the

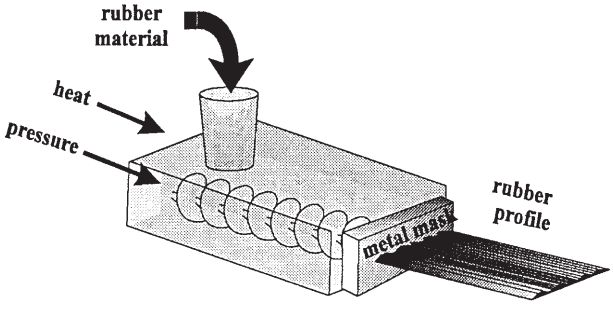

Fig. 1. The tyre profile extrusion is done by heating up a rubber mixture in a machine, called an extruder, until it becomes liquid, and then pressing it through a small opening, called a metal mask. The resulting rubber stream expands and solidifies after cooling down in the open air.

Correspondence and offprint requests to: Dr R.W. Brause, J.W. Goethe-University, FB Informatik, Postbox 11 19 32, D – 60054 Frankfurt am Main, Germany. E-mail: brause@cs.uni-frankfurt.de

time taken for 2–3 turn-arounds, each taking 4–5 days to make a new mask, install it onto the extruder, attempt an extrusion, measure the rubber profile obtained, and estimate for a more accurate metal mask.
- This delay not only wastes time, money and natural resources, but increases the production overheads and so impedes production flexibility severely.
- Experienced employees are tied to this job (which is judged as 'boring'), without the possibility of a change within the company.
- In the case of employee illness or an employee moving to another enterprise, the experience is no longer accessible.

This kind of problem can now be overcome by adaptive process control methods. Generally, these methods are applicable in one of the following situations:

- The process control is analytically unsolved.
- In principle, the problem can be solved analytically, but it is too expensive to do this for every case, or there are no qualified people available to do this.
- In principle, the problem can be solved analytically, but the necessary internal parameters of the process cannot be measured because either the measurements will change the values themselves, or the measurement is either technically infeasible, too difficult or too expensive.

Adaptive methods will update the parameters based purely on the final, measured outcome data. In this paper, we show how this sort of approach can be applied to the problem of tyre production.

Alternative, non-adaptive approaches do not exist in this field, because here the exact solution depends upon knowledge about the non-linear behaviour of the rubber, which is not currently available. Also, models using volume-oriented rubber flow within the tyre profile have not yielded any useful results.

In the following, we consider the problem of approximating the metal mask necessary by means of an adaptive system, i.e. an artificial neural network. The model for the network will be presented in the next section.

2. An RBF Approximation Network

In this section we want to derive an algorithm for approximating the exact extruder metal mask profile $f(x)$ at location x by a neural network function $F(x)$, which produces the desired rubber profile $r(x)$. It is well known that a two-layer neural network can approximate any continuous function to any degree, provided that we have enough neurons in the first layer [1].

2.1. The Activity Network

For our purposes, let us assume a two-layer network, like the one in Fig. 2. The network receives as input n signals, grouped together to the input $\mathbf{x} = (x_1, \ldots, x_n)$.

The activity $\mathbf{y} = (y_1, \ldots, y_m)$ of the first layer of units is defined by

$$y_i = S_i(\mathbf{x},\mathbf{c}_i) = \mathrm{e}^{\frac{(\mathbf{x}-\mathbf{c}_i)^2}{2\sigma_i^2}} \quad i = 1 \cdots m \tag{2.1}$$

and the second layer by

$$F(\mathbf{x}) = F(\mathbf{y}(\mathbf{x})) = \sum_{j=1}^{m} w_j y_j = \mathbf{w}^{\mathrm{T}}\mathbf{y} \quad y_0 \equiv 1,\ w_0 \cong \text{bias} \tag{2.2}$$

This models the approximation function $F(\mathbf{x})$ as a linear superposition (weighted sum) of m nonlinear basis functions S_i, which depend only upon the distance between the input $\mathbf{x}$ and a centre $\mathbf{c}_i$:

$$S_i(|\mathbf{x} - \mathbf{c}_i|) = S_i(D) \quad D^2 = (\mathbf{x} - \mathbf{c}_i)^2/2\sigma_i^2$$

This proportion gave them the name Radial Basis Functions (RBF).

2.2. Scaling the Receptive Field

Often, the different signals x_i in $\mathbf{x} = (x_i, \ldots, x_n)$ have different variances. Before we can combine them, we have to normalise them to equal variance in order to balance the different scaling influences of their origins. For this purpose, we scale the input space using a linear transformation with a matrix $\mathbf{M}$. For the set of input patterns, this is often done by replacing the ordinary Euclidean distance $D^2 = (\mathbf{x} - \mathbf{c})^2$ by the Mahalanobis distance

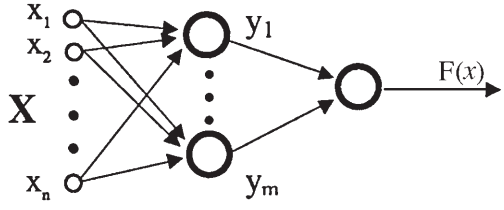

Fig. 2. The activity approximation network. The input lines are processed by special RBF-units. Their outputs are linearly weighted and added together to form the network output. All units with the same function are grouped as a layer, thus we have two layers: m RBF-units in the first layer, and one unit as the second layer.

$$D^2 = |\mathbf{M}(\mathbf{x}-\mathbf{c})|^2 = (\mathbf{x}-\mathbf{c})^{\mathrm{T}}\,\mathbf{M}^{\mathrm{T}}\,\mathbf{M}(\mathbf{x}-\mathbf{c}) \tag{2.3}$$

which is related to the covariance matrix of the input by

$$\mathbf{C}_{xx}^{-1} = \mathbf{M}^{\mathrm{T}}\,\mathbf{M}$$

If we assume an input activity to be significant for a neuron if it passes a certain threshold value θ, we might define the *receptive field* $\mathcal{R}_F$ of the neuron as the set of all inputs for which the neuronal output activity S passes the threshold

$$\mathcal{R}_F \equiv \{\mathbf{x} \mid S(\mathbf{x}) > \theta\}$$

The border $\{\mathbf{x} \mid S(\mathbf{x}) = \theta\}$ of the receptive field, and therefore its form, is determined by the parameters M_{ij}. If $\mathbf{M}$ is the identity matrix, the receptive field is the unit circle. For non-diagonal matrices the boarder becomes ellipsoidal.

For the pure purpose of designing the form, let us develop a scaling algorithm for the matrix $\mathbf{M}$. Let $\mathbf{x}$ be a sample input and $\mathbf{c}$ the centre of a distribution (see Fig. 3).

The new vector $\mathbf{x}'$, which is scaled relative to the centre $\mathbf{c}$, is obtained by reducing $\mathbf{x}$ with a fraction β of its radial component. With the distance vector $\mathbf{D} = (\mathbf{x} - \mathbf{c})$ and the normalised version $\mathbf{a} = \mathbf{D}/|\mathbf{D}|$, this becomes

$$\begin{aligned}\mathbf{x}' &= \mathbf{x} - \beta\mathbf{a}\,(\mathbf{a}^{\mathrm{T}}\mathbf{x}) = \mathbf{x} - \beta(\mathbf{a}\mathbf{a}^{\mathrm{T}})^{\mathbf{x}} \\ &= (I - \beta(\mathbf{a}\mathbf{a}^{\mathrm{T}}))\,\mathbf{x} = \mathbf{A}\mathbf{x}\end{aligned} \tag{2.4}$$

The change factor $\beta \in [0,1]$ (zero for no change, one for complete radial reduction) can also be replaced by a scaling factor $\alpha \equiv 1 - \beta$ (i.e. $\alpha = 0$ for complete radial reduction, $\alpha = 1$ for no reduction at all, $\alpha > 1$ for scaling up). A learning rate factor γ can also be included. For the basis vectors of the transformation (2.3) (which are the rows of $\mathbf{M}$) the scaling Eq. (2.4) becomes

$$\begin{aligned}&(\mathbf{M}^{\mathrm{NEW}})_j = (I - \gamma(1-\alpha)(\mathbf{a}\mathbf{a}^{\mathrm{T}}))\,(\mathbf{M}^{\mathrm{OLD}})_j \\ &\text{for } j = 1..k\end{aligned} \tag{2.5}$$

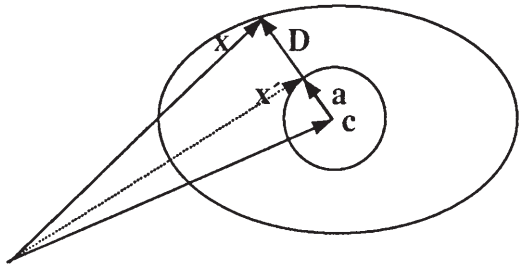

Fig. 3. The scaling of an input relative to the centre. For the scaling, the vector **D** from the centre **c** to **x** is scaled down to a vector **a** in the same direction, but of length 1. This results in a radially symmetric receptive field, instead of an ellipsoidal one.

2.3. The Learning Algorithm

There are principally two approaches to train the network parameters: we either train the two layers separately or as a whole. Each of these approaches has its advantages and its flaws, which we discuss briefly in order to develop our learning algorithm.

The first choice is to treat the two layers separately. This avoids the well-known local minima and the very slow speed which we encounter using the classical backpropagation algorithm [2] for two layers. A common approach consists of first clustering the input space using a learning algorithm (e.g. either off-line clustering using the k-means algorithm [3] or an RBF version [4], or on-line sequentially using an adaptive version [5] or an RBF version [6]). This sets up the number of neurons, the centres $\mathbf{c}_i$ of the receptive fields and their distance matrix $\mathbf{M}_i$. After this, the weights w_j of the second layer can be learned by a purely gradient descent. Since we have only a linear neuron in the second layer, there is only one minimum for the mean squared error.

This approach is fast, but it has some flaws:

- This input sample density generally does not correspond to regions where the approximated function changes quickly. This gives us a high sample density of output values, where we have clusters of input samples, not where the output error is high.
- The approach of homogeneously covering the input space by neuronal receptive fields based on clustering is not appropriate for all approximated functions. Consider, for instance, the function

  $$f(x) = \mathrm{e}^{-(x-a)^2} - \mathrm{e}^{-(x-a)^2/10}$$

  As we can see, two RBFs, both centred on centre a, will certainly approximate $f(x)$:

  $$F(x) \equiv S_1(x,a,1) - S_2(x,a,1/10)$$

  But this is not possible in the separate layer clustering approach above, because their different neurons get different cluster centres which together cover the input space. Therefore, we need many more neurons to approximate $f(x)$ and get less precision.

These problems lead us to the approach of optimising both layers at the same time. To avoid the computational problems of the backpropagation approach, we choose the strategy of starting with the lowest possible complexity of the network and gradually increase the number of neurons in the first layer until the error is sufficiently reduced. The resulting network will fit the approximation needs with the least possible resources. The ‘growing net-

work' approach has already been proposed for Kohonen nets by Fritzke [7] and for RBF nets, for example, by Schiøler and Hartmann [8].

There is also the possibility of starting with a very high number of neurons covering the input space, and then gradually pruning the network by eliminating all unnecessary neurons [9]. This approach is less favourable, since it has some principal flaws:

- In some cases, the increasing complexity approach may produce very high neuron densities at some points of the input space. To obtain the same densities (i.e. error coverage) using the pruning method, we have to start first with a very dense grid and then prune all the unnecessary RBF neurons using the algorithm. This gives a heavy computational load compared to our approach, because our approach deals only with necessary neurons, not with unnecessary ones.
- There is a visualising technique for the pruning process which gives some ideas about when and where neurons are eliminated by drawing the neuron positions in the input space. This kind of visualisation is not possible for our application, discussed in Section 3, since there we have an input space of $k = 11$ dimensions, which can hardly be visualised.

Therefore, we did not consider the pruning technique.

Let us discuss our approach in more detail. For our application (as for most of the control applications), we have to reduce the maximal possible error, not the Mean Squared Error (MSE). Therefore, we have devised a new strategy which is different to those mentioned before. We insert the first neuron (and all following ones) at location $\mathbf{x}_k$, the kth sample, with the *maximal* error

$$|f(\mathbf{x}_k) - F(\mathbf{x}_k)| = \max_1 |f(\mathbf{x}_i) - F(\mathbf{x}_i)| \tag{2.6}$$

The error

$$z_m(\mathbf{x}_k) = f(\mathbf{x}_k) - F(\mathbf{x}_k) = f(\mathbf{x}_k) - \sum_{j=0}^{m-1} w_j S_j(\mathbf{x}_k) \tag{2.7}$$

have to be compensated by the new mth neuron

$$w_m S_m(\mathbf{x}_k) = z_m(\mathbf{x}_k) \tag{2.8}$$

Because we insert neuron m at location $\mathbf{c}_m = \mathbf{x}_k$, we have $S_m(\mathbf{x}_k) = 1$, and therefore Eq. (2.8) gives us the value for the weight

$$w_m = z_m(\mathbf{x}_k) \tag{2.9}$$

The only unspecified parameter for the neural net is the width of the receptive field of neuron m, characterised by $\mathbf{M}_m$. The width should be designed so that it fits the new basis function in the context of all neighbouring neuron basis functions.

Initially, we try to fit the output activity contribution z_m such that it also remains favourable for the neighbouring training data points $\mathbf{x}_i$, i.e. it does not increase the error

$$|z_{m-1}(\mathbf{x}_i)| \geq |z_m(\mathbf{x}_i)|$$

With $w_m S_m(\mathbf{x}_k) = z_m(\mathbf{x}_k)$, and Eq. (2.1) we get $S_m = \mathrm{e}^{-D^2} = (z_{m-1})/(w_m)$, or

$$D^2 = |\mathbf{M}_m(\mathbf{x}_i - \mathbf{c}_m)|^2 = -\ln \frac{z_{m-1}}{w_m} = \ln \frac{w_m}{z_{m-1}} \tag{2.10}$$

Let us initialise $\mathbf{M}_m$ by the scaled unity matrix $\alpha\mathbf{I}$, i.e. we assume a scaled circular receptive field of radius α. For the first neighbour, we have

$$D^2 = \alpha|(\mathbf{x}_i - \mathbf{c}_m)|^2$$

which gives us, with Eq. (2.10), the scaling factor

$$\alpha = \frac{D}{|\mathbf{x}_i - \mathbf{c}_m|} \tag{2.11}$$

This is straightforward if the function value $f(\mathbf{x}_i)$ of the neighbour input sample has the same sign as $f(\mathbf{x}_m)$ on the new neuron's location. When they have different signs, the situation changes: we can no longer adapt the receptive field directly, because principally it cannot change the sign by adaptation. Instead, by inserting a replacement point $\mathbf{x}$. which lies on the distance between $\mathbf{x}_i$ and $\mathbf{x}_m$ and has a linear interpolated value of the same sign, we design the receptive field for a sharp decrease to become approximately zero at $\mathbf{x}_i$ [10].

How should we treat the other neighbouring data points? In contrast to the approach of Platt [11], we do not use the gradient descent technique to rearrange all the other neurons and adapt all their receptive fields, which is computationally intensive and is the source of new errors. Instead, we might stop the adaptation process of the new neuron using some criterion. Here, we have several possibilities:

1. We might look for the maximal error of all neighbouring points, and if we do not reduce the error by an adaptation, we should stop. In simulations this strategy produced unnecessary errors, because neurons inserted early on do not have well adapted receptive fields, which are too big and thus dominate the error amount.
2. We might stop if the output $F_{m-1}(\mathbf{x}_i)$ changes its sign compared to $F_{m-1}(\mathbf{x}_k)$. Here the contribution

z_m will not diminish but increase the error. Nevertheless, since we do not know whether there are other neurons far away with a greater error which can be compensated by z_m even though the error in the neighbourhood is increased. Thus, this is also not a good criterion.

3. We might consider only data which cause significant activation of the new neuron. Here, we stepwise decrease an activity level z and look for data which causes the neural activity to exceed this level:

$$z \leq z_s$$

For each neighbour, $\mathbf{M}_m$ is adapted according to Eqs (2.11) and (2.5) and the replacement point techniques, if necessary.

Simulation results support the third strategy as being the most effective one, so this was chosen to serve for the industrial application. Additionally, we reduced the long distance neighbourhood influence by a learning rate $\gamma(d)$ which drops with increasing distance from $\mathbf{c}_m$, i.e. with decreasing activity level.

The whole growing and initialisation learning algorithm, called GGRBF (growing generalised RBF), can be formulated in pseudo code. With the maximal tolerated Error `TolErr` and the maximal number $m_{\max}$ of neurons, we get

```
GGRBF:
m: = 0; Errset (Trainingset) : =f (Inputset);
WHILE ( max (Errset) >TolErr ) AND (m<m_max) DO
 x = coord (max (Errset))        (* location of maximal error*)
 InsertNeuron (x)                (*see strategy (3)*)
 AdaptTolstNeighbor (M_m);       (*(2.10) and (2.11)*)
 level : =1; γ: =1;              (*start with high activation level*)
 WHILE level> 0.01 DO
  FOR i: =1 TO |Trainings Set|    DO  (*strategy (3)*)
   IF S_m(x_i)>level THEN AdaptToNeighbor (γ, x_i, M_m) END
  ENDFOR
  γ: =γ*0.87;                    (*diminuate learning rate*)
  level: =level-inc;             (*lower the attention level*)
 ENDWHILE
 m: =m+1                         (*new neuron installed*)
 computeErrset (Trainingset);    (*⇒ new error landscape*)
ENDWHILE
```

2.4. Simulation Results

The performance of the GGRBF algorithm was tested on a number of functions. For visualisation purposes, we restrict the input space to two dimensions. In all the simulations we restricted the training set to 50 points and the net to 10 neurons.

One of the most difficult tasks for nonlinear RBF nets is a simple plane

$$f_1(\mathbf{x}) = x_1 + x_2$$

The approximation of the plane by $F(x_1,x_2)$ of a growing RBF algorithm with only a circular receptive fields (GSRBF) algorithm using only 10 neurons is shown on the left-hand side of Fig. 4, while on the right-hand side we plot the contour lines and the receptive fields $\mathcal{R}_F$ with $\theta = 0.1$.

The landscape of the function of the growing RBF algorithm with ellipsoidal receptive fields (GGRBF algorithm) for 10 neurons is shown in Fig. 5.

The timing performance evolution for the nets is shown in Fig. 6. Here, the time requirements for conventional gradient descent backpropagation algorithms (SRBF and GRBF algorithms) to adapt to F_1 are compared to the growing GSRBF and GGRBF algorithms with the same performance error.

Here, we can clearly see the advantage of starting with a low complexity. The gradient technique dramatically increases the computation time up to 100 times for the same approximation performance. This is also true for the other simulations.

To be fair, it should also be noted that in the case of function F_1, the gradient descent method allowed further enhancements of the approximation by additional training for the 10 neurons, whereas this is not possible for the growing net method.

2.5. Other Benchmarks

For the sake of comparison, let us consider as benchmarks the functions used by the growing net with the gradient descent method of Lee and Kil [12]. They reported that the function

$$f_2(\mathbf{x}) = \sin(\pi x_1) \cos(\pi x_2/2)$$

was approximated using 50,000 training samples. In Fig. 7, on the left-hand side we show the function, and on the right the simulation results.

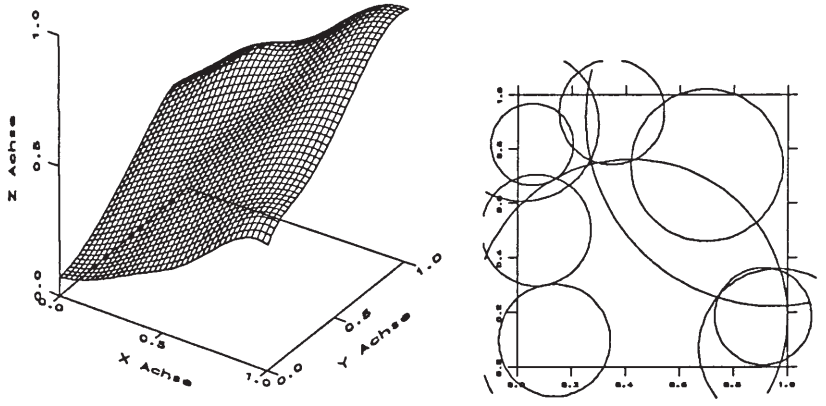

Fig. 4. The approximation of a plane by radially symmetric RBF. On the left-hand side we can see the approximated function $F(x,y)$, whereas on the right-hand side the borders of the receptive fields of the 10 neurons are shown in the $x - y$ plane. The linear behaviour of the surface is obtained by the overlay of differently sized receptive.

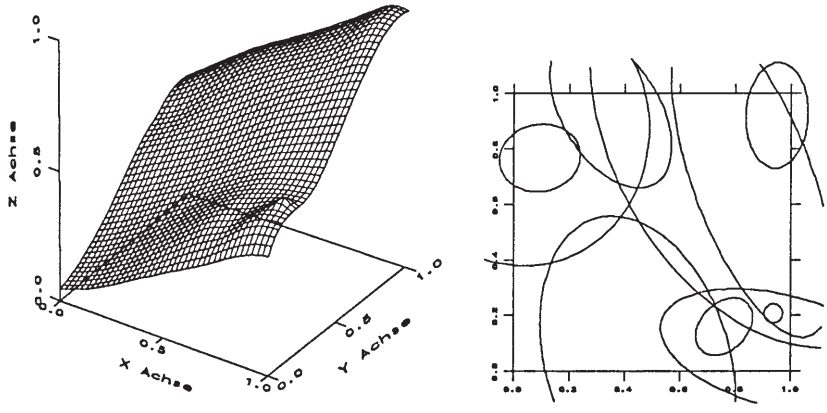

Fig. 5. The approximation of a plane by ellipsoidal RBF. The analogue plot (as in Fig. 4) is shown here for ellipsoidal receptive fields. This time, the receptive fields are superposed more smoothly.

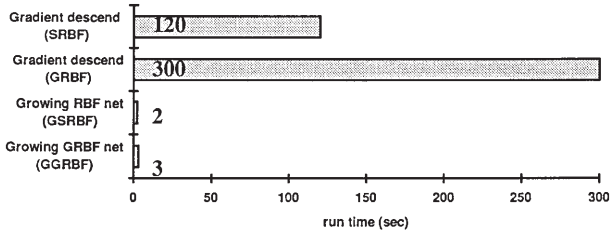

Fig. 6. The timing performance of the different algorithms. For the same performance, our growing nets improve much faster, because we do not have to correct all previous sample influences when adding a new unit.

The more complicated function

$$f_3(\mathbf{x}) = \cos(4\pi x_1)\cos(4\pi x_2)\, e^{-10(x_1^2+x_2^2)}$$

and the approximation performance is shown in Fig. 8.

In both cases, we can see that our growing network with local adaptation performs better than the network of Lee and Kil adapting using a gradient algorithm.

3. Approximating the Extrusion Process Parameters

To apply the approximation algorithm developed in the previous sections, we have to model the industrial process for the tyre production example. As described in the introduction, the main task consists of estimating the profile of a metal mask which extrudes the profile of a rubber band. This band is then cut into a strip with the perimeter length of a tyre, and then glued to the casing. The raw tyre is then 'baked' in a metal tyre form for 20 minutes, giving the preliminary profile the ultimate form.

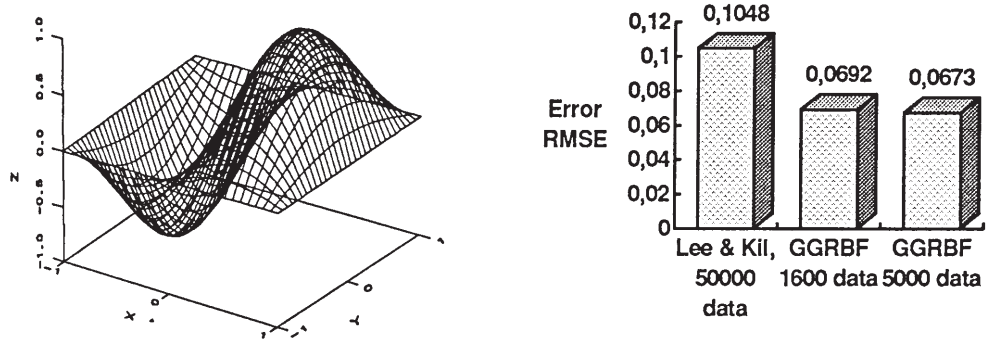

Fig. 7. The function f_2 and its different approximation performances. The smooth trigonometric function $f_2(x,y)$, shown on the left-hand side, is approximated by different means. The error plot on the right-hand side shows a better performance for the same number of neurons with 10 times less training data, and therefore also shorter algorithm run times.

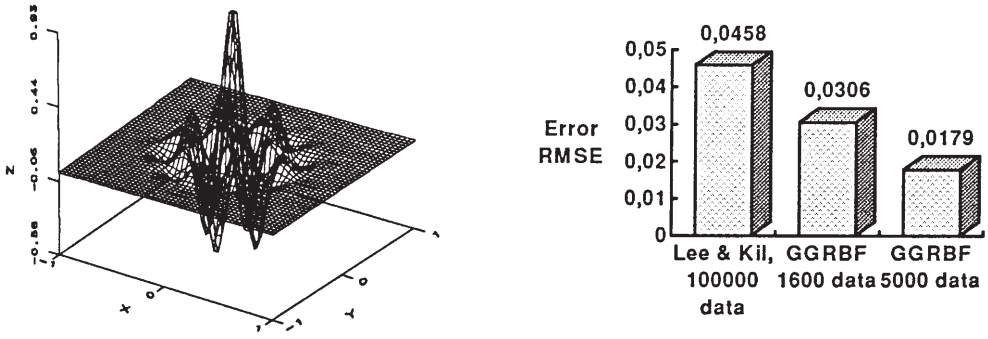

Fig. 8. The function f_3 and its different approximation performances. For the complicated trigonometric function $f_3(x,y)$, shown on the left-hand side, the growing net also outperforms the classic approach, obtaining less than half of the error by training with only 5% of the data.

3.1. Modelling the Process

Although the extruded rubber profile is a temporary form, its desired accuracy is 0.1 mm. This settles the upper limit for our approximation error. In Fig. 9 we show a sample profile.

The upper profile is the desired rubber profile, the lower one shows the corresponding rectangular metal mask. On the right-hand side a cut through the metal (shaded area) shows the form of the opening (not shaded). The profile has a wider opening where the rubber flows in. This corresponds to

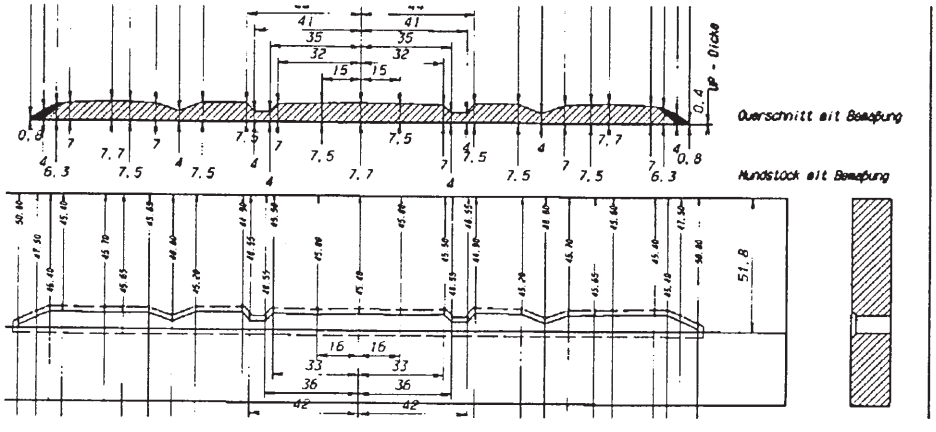

Fig. 9. A rubber profile and the corresponding metal mask. In the upper figure, a cut through the rubber band is shown, whereas in the lower figure the cutout of the metal mask bar is shown in the front view. The dotted line denotes the (carved) edges of the opening. This is also visualised in the vertical cut of the metal mask on the right-hand side.

the dotted line which encircles the profile opening in the metal mask.

The modelling has to reflect the following:

- The profile of the extruded rubber band principally depends upon the volume of extruded rubber. The rubber expansion pressure and flow within the profile depends greatly on whether there is 'a huge amount of rubber', i.e. the neighbouring parts of the profile have a high level, or if we have 'very little rubber around', i.e. the neighbouring parts are of a low level. This means the rubber profile is also a function of the profile height of the neighbouring points.
- Additionally, the extruded rubber profile heights depend nonlinearly upon the rubber mixture G, the pressure P from the screw conveyor, on the temperature T and on the extruder type E.
- Because of the nonlinear form of the screw conveyor, the pressure along the profile mask decreases nonlinearly. This depends upon both the extruder machine and profile type. Therefore, the rubber profile also depends upon the absolute position along the metal mask.

Nevertheless, the whole system is deterministic: the same rubber mixture G with the same mask $g(x)$, temperature T and pressure P result in the same rubber profile $r(x)$ on a different extruder machine of the same type E. The analytical treatment of the nonlinear dependencies is very difficult. Conventional assumptions about energy (i.e. enthalpy) conservation are not valid here. Also, the direct measurement of the process parameters like temperature and pressure in the profile are limited practically. The sensors have to be incorporated in such a way that they do not constitute an obstacle themselves, otherwise the pressure conditions will be changed and give different results. This is practically impossible, or at the least, very expensive.

In contrast, our approach models the system as a whole, avoiding all difference equations and constants which are hard to devise and measure. In particular, the model of a neural network with locally sensitive elements underlines the local character of the modelling. We devided the whole centred profile, depending on the tyre width, into 170–270 points, placed at a regular distance of d mm. Each point has a desired rubber profile height $r(i)$. Since the profile data initially contains only points of profile change $(x_1, r(x_1), x_2, r(x_2), \ldots)$, the intermediate points are generated by interpolation (Fig. 10).

Since the influence of the sample points is limited to the neighbourhood for a certain rubber profile height $r(i)$, we only have to consider $k = 2s + 1$ neighbouring points

$$g(i) = F(r_{i-s}, \ldots, r_i, \ldots, r_{i+s}, i, G, E, P, \ldots) \equiv F(x_i, \ldots, x_n)$$

Using this model, we implement a neighbourhood window which uses $k = 2s + 1$ sampling points around location i. All values k_k for the sampling points outside the profile limits are set to zero.

3.2. Simulation Results

An important key for the simulation performance turned out to be the two parameters k, the number of neighbourhood sampling points, and d, the distance between the sampling points. The proper choice is determined by balancing the counter-acting influences:

- If we choose d too small, we increase the number of necessary sampling points for a certain neighbourhood and thus increase the dimensions of the input space. Since we have only a limited number of training samples, the training becomes very difficult, since the input space becomes very sparse. On the other hand, if we choose d too large, important information can be lost due to undersampling of the dependency function.
- If we choose k too large, we encounter the same problem of dimension inflation and training difficulties due to the sparseness of the training samples in the input space. Additionally, by increasing the context information too much, the generalisation ability of the network will be limited. On the other hand, if we limit the window too much, necessary context information which helps to distinguish between different situations is ignored, resulting in an unnecessarily randomised training.

From a theoretical point of view, this is an interesting situation. Nevertheless, we are not aware of any applicable method to determine the optimal d and k to solve the problem of optimal training. Therefore, we decided to simulate different configurations in order to get an acceptable choice for the parameters.

We generated the training set by shifting a window (determined by d and k) by an increment of 1 mm over the profile data of five profiles with the same values of G, E and P. This generated 1346 training patterns. The sixth profile was used for the generation of a test set of 271 test points. Our multi-dimensional approximated function became

$$g\,(i) = F(r_{i-s}, \ldots, r_i, \ldots, r_{i+s}, i, w)$$

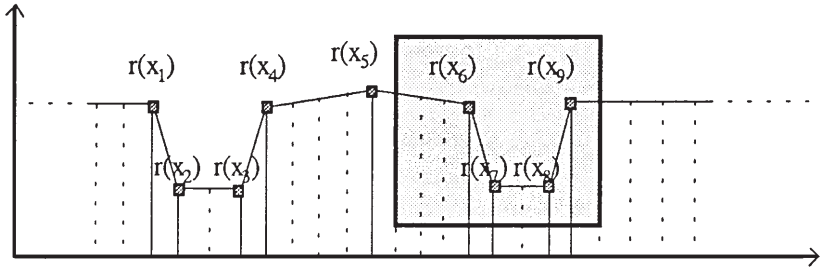

Fig. 10. The intermediate interpolation of the profile. The intermediate points are obtained by equidistant points and are denoted by dotted lines. This procedure coverts the profile into a fixed raster of constant point number size, whereas the initial representation minimises the storage requirements by the variable number of profile points and inter-point distances.

with w being the weight per meter of the extruded rubber band. The simulation results generally showed only a very small influence of the position i. So, let us consider other dependencies.

For the expected absolute error for 100 neurons we obtained different results, depending on the type of network we used. Generally, the GGRBF nets are more successful than the GSRBF nets. The GSRBF nets with growing, radially symmetric input regions have, on average, 10–90% more errors than growing ellipsoidal nets. The best performance of the two types converged by training to the following expected absolute error, depending on the number of sampling points k and the interpoint distance d. In Fig. 11, this is shown for $k = 7,9,11$ for each of the intersample distances of $d = 3,4,5$ mm.

It is interesting to see that the error does not automatically decrease when we increase the number of sampling points. There is a configuration of the parameters where the balance is roughly met and the error becomes quite small.

The best results are observed by $k = 9$ and $d = 4$ mm, which corresponds to a window size of

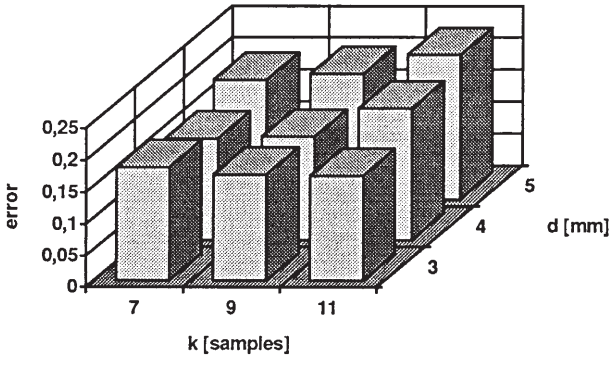

Fig. 11. The error development for different parameter values of d and k. The expected absolute error of the approximation depends, in a non-linear manner, upon the inter-sample distance d and the number k of samples used for approximating one profile point. For $d = 4,5$ the error also increases when k is increased due to long distance disturbance influences for a fixed number of units, i.e. iterations.

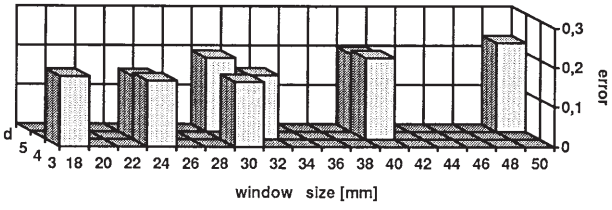

Fig. 12. The error development and window size. The data of Fig. 11 can also be interpreted in the light of the density of the sample points. Therefore, the nine results are plotted for the window size $w = (k - 1)d$. In the plot derived, we see that above a certain level, increasing the window size also increases the error monotonously.

32 mm. In Fig. 13, the test profile, the result of the network and the resulting error is shown for this configuration. The expected absolute error was 0.16 mm, the maximal absolute error 0.56 mm. The y-axis is scaled up by the factor of three to enhance the visibility of the errors.

Why is there still such a big error? For example, let us consider the centre. When we scale up the error, the drawing in Fig. 14 arises. Here, the typical

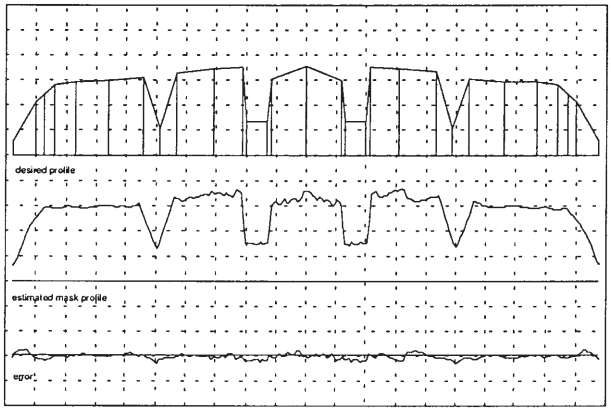

Fig. 13. The desired profile and the profile produced by the net for $k = 9$, $d = 4$. For visualisation purposes, the profiles and the error is multiplied three times. You can clearly see that the error is particularly increased in the neighbourhood of strong changes in the profile.

influence of the sampling window is shown. The size of the sampling window is 32 mm, whereas the width of the profile hill is 34 mm. Since in the training, on average, there are no valleys on the right and left-hand sides, the net 'assumes' more rubber volume on the right and left hand sides which will bring up the middle. Here this is not the case, so without the anticipated neighbouring rubber pressure, the top in the middle is not reached and an error occurs.

To get rid of this effect, we have to enlarge the window and include the neighbouring information about the neighbouring lack of rubber material. If we do this, the error will also increase. Why? This can be explained by the sparse input space: if we enlarge the input space without filling it with training patterns, the whole system learns less. This problem is known as the 'curse of the dimensions' [13]. The only remedy for this is the augmentation of the number of training patterns.

4. Discussion and Outlook

In the previous sections, we presented an adaptive solution for the problem of unknown process parameters in tyre production. The proposed neural network learns the function which estimates the form of the metal profile for the extrusion of a rubber band when the rubber profile is given as a goal. The learning algorithm uses no internal process variables or other intrinsic knowledge, but only the measurable external process parameters, such as the weight per meter and the resulting rubber profile.

This approach has many technical advantages:

- There is no intrinsic system knowledge necessary, like non-linear dependencies or differential equations for modelling.
- The same adaptive program can be used even when the parameters change, due to a change in the non-modelled system background context.
- There is the theoretical possibility to obtain for a new rubber mixture all the necessary estimation parameter values just by training with one standard profile. By using a generalised adaptation, the initial parameters will determine the correct metal mask for all possible desired profiles.

Nevertheless, our work also shows that there are still several problems to be solved:

- The current modelling uses the data provided by the production in the form of tuples (desired rubber profile, successful metal profile). The successful metal profile was obtained after several trials and corrections. Since the trial-and-error cycle was stopped when the 'overall' error was small enough, this results not in a good training profile, but in an error deviated training profile.

  The way in which to solve this problem is easy: as training samples, we have to use the directly laser-measured profiles of the metal mask and the resulting rubber profile, even if the rubber profile is not the desired one. This involves an additional data measuring stage and the corresponding software that is necessary.
- The number of training profiles is not high enough to contain the information necessary for the determination of all parameters.

This problem is not easy to solve. By the very nature of the problem, we do not have hundreds of sample profiles, just a few. The careful dimension analysis (necessary neighbour points) will help us to obtain the balance between input dimension, learning complexity and the number of neurons and parameters. Here, the work of the theorist is welcome.

An interesting alternative approach for computing k and d is the application of evolutionary algorithms, which search for the parameter optimum of k and d by random strategies [14]. Whereas this approach can potentially give good results, it is computationally heavy and does not provide any new insights into the underlying structure.

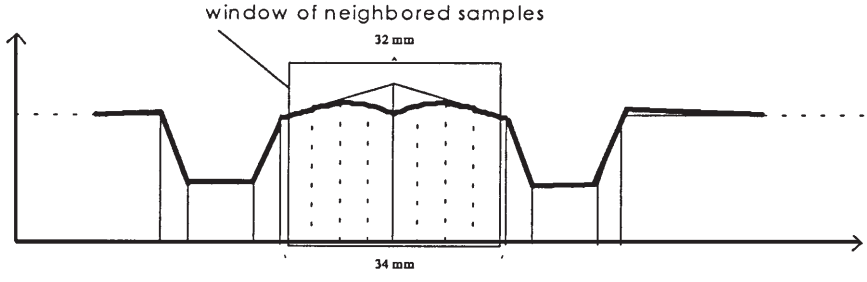

Fig. 14. The error and the sampling window size ($k = 9$, $d = 4$). When the sampling window is too small, the deep ridges on the left- and right-hand sides are not seen by the system, and the prediction assumes too much rubber volume flow from the sides to the middle. This results in a faulty estimation of the necessary metal profile in the middle.

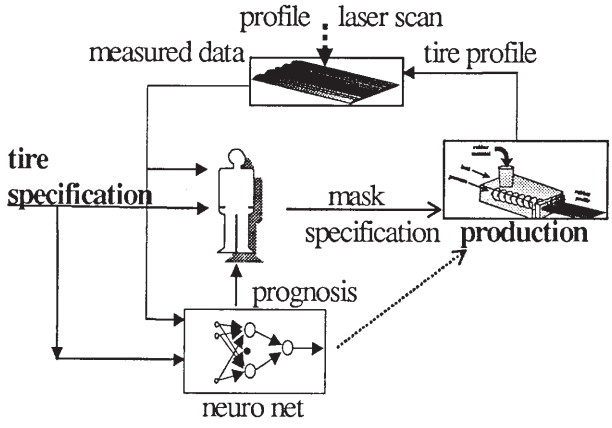

Fig. 15. The adaptive control for process parameter estimation. Initially, the adaptive loop consists of a human being, estimating the mask profile properties due to a given tyre profile specification. After producing a small number of rubber profiles, the profiles are measured, compared to the specification and the obtained error is corrected by correcting the metal mask. This loop is performed several times, updating the implicit knowledge of the estimation by the human operator. In this adaptive loop, the human estimation is replaced by the neural network.

The introduction of automated estimation in the fabrication process must be carefully planned in order to be accepted by employees. The adaptive process control scheme shown in Fig. 15 uses the neural network control as a bypass for the human-based estimation process. As soon as human operator confidence in the software is high enough, he or she automates the transfer of the network results to the profile mask cutting device.

The economical and human labour context implications are discussed more deeply in a separate publication [15].

References

1. Hornik K, Stichcombe M, White H. Multilayer feedforward networks are universal approximators. Neural Networks 1989; 2: 359–366
2. Rumelhart DE, McClelland L. Parallel Distributed Processing. Vols I,II,III. MIT Press, Cambridge, MA 1986
3. Tou JT, Gonzalez RC. Pattern Recognition Principles. Addison-Wesley, Reading, MA, 1974
4. Musavi WA, Chan K, Faris K, Hummels D. On the training of radial basis function classifiers. Neural Networks 1992; 5: 595–603
5. Kohonen T. Self-Organisation and Associative Memory. Springer-Verlag, Berlin, 1984
6. Xu L, Krzyzak A, Oja E. Rival penalised competitive learning for clustering analysis, RBF Net, and curve detection. IEEE Trans Neural Networks 1993; 4(4): 636–649
7. Fritzke B. Growing cell structures – a self-organizing network in k dimensions. In I Aleksander, J Taylor (eds), Artificial Neural Networks II, North-Holland, 1992, pp 1051–1056
8. Schiøler H, Hartmann U. Mapping neural network derived from Parzen window estimator. Neural Networks 1992; 5: 903–909
9. Hong X, Billings SA. Given rotation based fast backward elimination algorithm for RBF neural networks pruning. IEE Proc Control Theory Applic 1997; 144(5)
10. Pietruschka. Funktionsapproximation mit RBF-Netzen. Diplomarbeit an der JW Goethe-Universität, FB Informatik, Frankfurt a.M., 1995 (in German)
11. Platt C. Learning by Combining Memorisation and Gradient Descent. NIPS, 1992, pp 714–720
12. Lee S, Kil R. A Gaussian potential function network with hierarchicallly self-organizing learning. Neural Networks 1991; 4: 207–224
13. Huber PJ. Projection Pursuit. Ann Statistics 1985; 13(2): 435–475
14. Dasgupta D, Michalewicz Z. (eds) Evolutionary Algorithms in Engineering Applications. Springer-Verlag, New York 1997
15. Brause R, Pietruschka U. Adaptive control in rubber industry. Int Occ Safety and Ergonomics (JOSE), Ablex Publishing 1998; 4(3): 253–269

Notation

$\mathbf{x}$	input vector of neural network
$\mathbf{y}$	output vector of the RBF-units in the neural network
$\mathbf{c}$	fixed parameter of a RBF-unit.
$S_i(\mathbf{x},\mathbf{c})$	output function ('activation function') of RBF-unit i
σ_i	variance parameter of the RBF-unit
W_j	weight from first layer to output unit ('second layer')
$z_m(\mathbf{x}_k)$	activity level in the output unit due to the kth input component of the mth sample
$F(\mathbf{x})$	scalar output function of the network
$f(\mathbf{x})$	teacher-related, unknown output function. The function values $f(\mathbf{x}_k)$ are the desired output values of the net
$\mathbf{D}$	distance between a input sample $\mathbf{x}$ and the centre of the neuron $\mathbf{c}$
$\mathbf{M}$	matrix for scaling the receptive field
α,β	scaling constants for the receptive field
γ	learning rate
$\mathcal{R}_F$	receptive field (set of input points)
θ	threshold used as lower limit of receptive field of a unit
$r(x)$	scalar function of the desired rubber profile
$g(x)$	scalar function of the estimated metal mask profile
k	number of samples used to estimate one profile point
n	number of input components of the network
d	scalar distance between two profile sample points