submitted to International Journal of Occupational Safety and Ergonomics, JOSE, 1998

Title: Adaptive Process Control in Rubber Industry

Authors: Rüdiger W. Brause , Ulf Pietruschka

Affiliation: J.W. Goethe-University, Frankfurt a.M., Germany

Short Title: Adaptive Process Control

Name and address of correspondence:

Dr. Rüdiger W. Brause

J.W. Goethe-University,

FB Informatik, NIPS

Robert-Mayer-Str. 11-15

Postbox 11 19 32

D - 60054 Frankfurt am Main

Tel.: +49 (69) 798 -23977

Fax: +49 (69) 798 -28353

Email: brause@informatik.uni-frankfurt.de

Abstract

This paper describes the problems and an adaptive solution for process control in rubber industry. We show that the human and economical benefits of an adaptive solution for the approximation of process parameters are very attractive.

The modeling of the industrial problem is done by the means of artificial neural networks. For the example of the extrusion of a rubber profile in tire production our method shows good results even using only a few training samples.

1 Introduction

In many industrial processes, the optimization of the process might reduce environmental damages and increase employee satisfaction as well as economical benefits. Here, the key problem lies in the optimal choice of the process parameters which are typically not available due to several reasons. This contribution tries to show that adaptive methods are important alternatives to conventional solutions.

Let us study this idea in more detail for a concrete type of production: the rubber industry.

1.1 A rubber industry production problem

Process control in rubber industry has the smell of a „dirty“ industrial branch. This relays not only on the often very dull and dusty rubber and tire production rooms where the products are „baked“ by heat and steam, but also on the fact that the macromolecular proportions of rubber are hard to predict due to their nonlinear character. When a rubber mixture leaves the extruder (the melting and form-giving machine) after being heated up to 110°-140°C, compressed to 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, changing therefore its shape in a non-linear manner by 10%-20% up to 50%.

The basic production layout is shown for our example of tire profile production in figure 1.

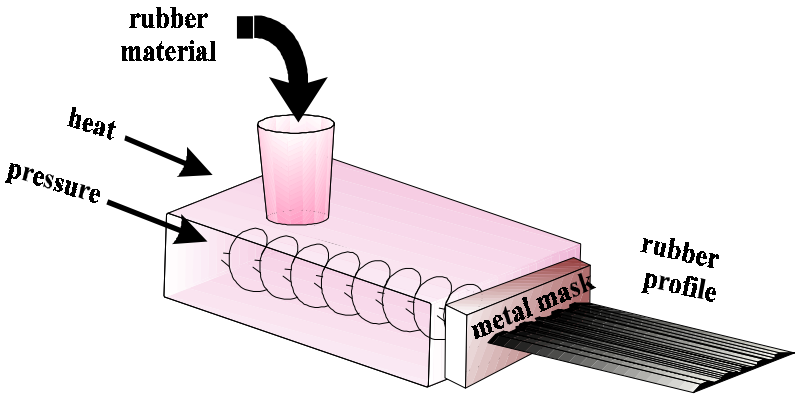

Fig. 1 The tire profile extrusion is done by heating up a rubber mixture in a machine, called an extruder, until it becomes liquid and then pressing it through a small opening, called a metal mask. The outcoming rubber stream expands and solidifies after cooling down at the open air.

The task of process control consists of estimating the necessary extrusion parameters (in our case: the unknown best shape of the extrusion metal mask) for an acceptable rubber product after relaxation. Up to now, due to the nonlinear nature of the macromolecular mixture this task can not be solved analytically. Instead, specialized people estimate the profile of the original metal mask by their experience with the subject and correct their estimates after experience. This gives a trial-and-error turn-around production cycle.

This kind of production causes severe disadvantages for the production business. For the management:

- the start for a new product is delayed by the time for 2-3 turn-arounds, each one taking 4-5 days to make a new mask, install it on the extruder, make an extrusion try, measure the obtained rubber profile and estimate a new metal mask,
- this delay does not only wastes time, money and natural resources, but also increase the production overhead and impedes therefore the production flexibility severely,
- in the case of illness of an employee or a change to another enterprise, the knowledge is no longer accessible. This causes mayor obstacles for the production.

Also from the employee's point of view, this kind of production is not satisfactory:

- The experienced employees are tied to this job (which is judged as „boring“) without the possibility of a change within the enterprise. The only possibility for a job change is to change to another enterprise which is often not possible due to many economical and personal reasons. Thus, motivation and identification with the job is severely impeded.

In conclusion, there are very attractive benefits of a possible solution to this problem, both on the economical and the human labor side.

1.2 Adaptive Process Control

Now, this kind of parameter estimation problems can be overcome by *adaptive process control* methods. Generally, these methods are interesting in one of the following situations in industrial process control:

- The process control is analytically *not* solved. So, until now only crude human estimations or outdated practices are available,
- In principle the problem can be solved analytically, *but* it is too expensive, or there is not enough time to do this for every variant of the problem, or there are no qualified people available to do this,
- In principle the problem can be solved analytically and there are people ready and time available to do this, *but* the necessary internal parameters of the process can not be measured because either the measuring device will influence the process and therefore the measurements itself, or the measurement is technically not feasible, or it is too difficult, or it is too expensive.

Contrary to the exact analytical solution, the adaptive methods will only update the parameters by an iteration process based purely on the final, measured outcome data. Adaptive control methods are used in many industry branches, especially for system control and identification, see (White 1992) and (Narendra 1986).

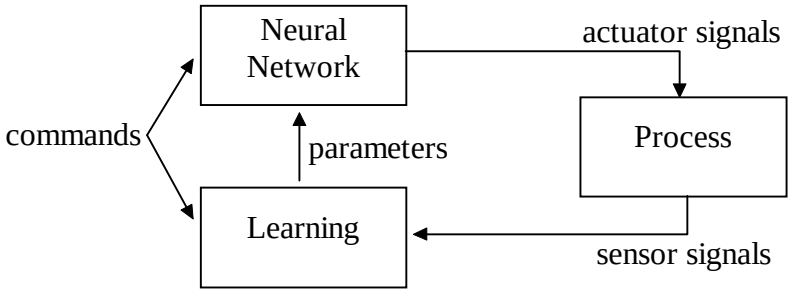

Fig. 2 The main layout for adaptive control. The control commands are mapped to the actuator signals (e.g. to pumps, levers and motors) by a neural network. The parameters controlling the network itself are updated by an adaptive algorithm which uses the results of the actuator signals (the sensor signals) and the control commands as input. The preprocessing of the sensor signals and the learning may also be done by extra specialized neural networks.

This method is often used in conjunction with the paradigm of Artificial Neural Networks, see e.g. (Haykin 1994) and Fuzzy Control (Buckley 1992).

In this paper we will show how this kind of approach can be done for the concrete problem of tire production. In the following we consider the problem of approximating the necessary shape of the metal mask by the means of an artificial neural network.

2 Approximating the extrusion process parameters

In order to apply adaptive parameter approximation algorithms we have to model the industrial process, in our example the tire production. As it is described in the previous section, 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 stripes of the perimeter length of a tire and

then glued to the casing. The raw tire is then „baked“ in a metal tire form for 20 minutes, which gives the preliminary profile its ultimate form.

2.1 Modeling 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. 3 a sample profile is shown.

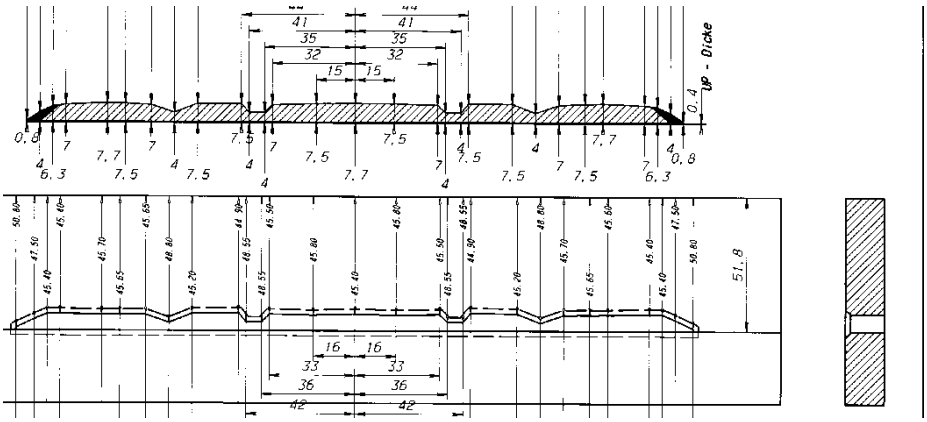

Fig. 3 A rubber profile and the corresponding metal mask. In the upper figure, a cut through the rubber band is shown, whereas in the lower figure the cutout of the metal mask bar is shown in the front view. The dotted line denotes the (carved) edges of the opening. This is also visualized in the vertical cut of the metal mask on the right hand side.

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

The modeling has to reflect the following facts:

- The profile of the extruded rubber band principally depends on the volume of the extruded rubber. The rubber expansion pressure and flow within the profile heavily depends on whether there is „a huge amount of rubber“ , i.e. the neighbor parts of the

profile have a high level, or if we have „very few rubber around“, i.e. the neighbor parts are low-leveled. This causes the rubber profile to be also a function of the profile height of the neighbor points,

- Additionally, the extruded rubber profile heights depend nonlinearly on the rubber mixture composition G, pressure P by the screw conveyor, temperature T and extruder type E,
- By the nonlinear form of the screw conveyor the pressure along the profile mask (along the axis of abscissae in fig. 3) decreases nonlinearly from the right to the left. This depends on the extruder machine type and on the profile type. Therefore, the rubber profile height does also depend on the absolute position on the axis of abscissae of 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

$$g(x) \xrightarrow{G,\ T,\ P,\ E} r(x)$$

The analytical treatment of the nonlinear dependencies is very difficult. Conventional assumptions about energy (i.e. enthalpy) conservation are not valid here, because the system is not closed and shows non-linear molecule behavior. Also the direct measurement of the process parameters like temperature and pressure in the profile are practically limited. The sensors have to incorporated in such a way that they do not constitute obstacles themselves, otherwise the pressure conditions will be immediately changed and give different results. This is practically impossible or very expensive.

In contrast to this, our adaptive approach models the system as a whole, avoiding all differential equations and constants which are hard to devise and to measure. Especially the model of a neural network with only locally distributed input for each neuron underlines the local character of the modeling.

We devided the whole profile, depending on the tire width, into 170-270 points which are placed in the regular distance of δ mm. For the comparison of profiles with different width, the centers of the profiles were lined up and treated as the common middle point of the grid. Each point on the grid has a desired rubber profile height r(i). Since the profile data initially contain only points of profile change (x_1,r(x_1), x_2, r(x_2), ..) the intermediate points are generated by interpolation, see Fig. 4.

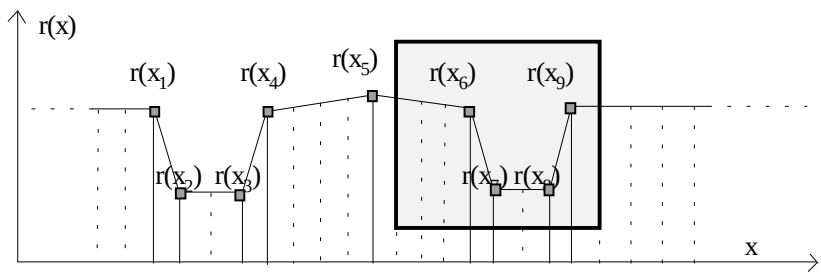

Fig. 4 The interpolation of the rubber profile. The intermediate points are obtained by equidistant points and are denoted by dotted lines.

Since the influence of the neighborhood is limited to a certain number s of sample points for a certain rubber profile height r(i) we have only to consider *k=2s+1* influencing sample points as parameters for the unknown metal mask function g(x)

$$g(x) = F(r_{i-s},...,r_i,...,r_{i+s}, i, G, E, P,..) \equiv F(x_1, ..., x_n)$$

In this model we implement a neighborhood window which uses k=2s+1 sampling points around location i. All values r_k for the sampling points outside the profile limits are set to zero.

3 The approximation network

In this section we want to derive an algorithm for approximating the exact extruder metal mask profile f(**x**) at location **x** = (x_1, ..., x_n) by a neural network, implementing the approximation function F(**x**). The metal mask will in turn produce the desired rubber profile r(**x**). It is well known that a two-layer neural network can approximate any continuous function to any degree of accuracy, provided that we have enough neurons in the first layer, see (Hornik, Stinchcombe, White, 1989) and (Xu, Krzyzak, Yuille, 1994).

3.1 The activity network

Now, for our purpose let us assume a two-layer network like the one in figure 5.

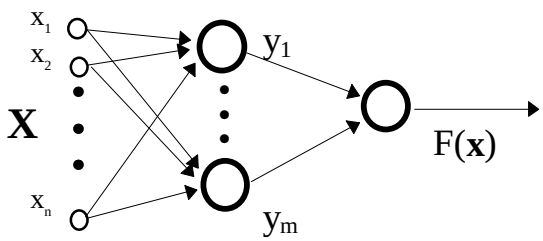

Fig. 5 The activity approximation network. The input lines **x**=(x_1,..,x_n) are processed by special RBF-units, denoted as black circles. Their outputs y_i are linearly weighted, added together and form the network output F(**x**). All units with the same function are grouped vertically in one layer. Thus, we have two layers: *m* RBF-units in the first layer and one unit as the second layer.

Each layer is composed of computational units called „neurons“. Each neuron computes a function S(**x,w**) of its input **x** and its parameters **w**.

Let us define the activity **y** = (y_1, ...,y_m) of the first layer by the activity of the i-th unit

$$y_i = S_i(\mathbf{x},\mathbf{c}_i) = e^{-d_i^2} \qquad i=1..m \qquad (1)$$

and the activity of the second layer by the linear function

$$F(\mathbf{x}) = F(\mathbf{y}(\mathbf{x})) = \sum_{j=1}^{m} w_j y_j = \mathbf{w}^T\mathbf{y} \qquad y_0 \equiv 1,\ w_0 \cong \text{‘bias’} \qquad (2)$$

This models the approximation function F(**x**) as a linear superposition (weighted sum) using *m* nonlinear basis functions S_i that depend only on the 'Mahalanobis distance' d_i between the input **x** and a 'neuronal center' $\mathbf{c}_i$.

$$S_i(|\mathbf{x}-\mathbf{c}_i|) = S_i(d_i) \qquad d^2 = |\mathbf{M}\,(\mathbf{x}-\mathbf{c})|^2 = (\mathbf{x}-\mathbf{c})^T \mathbf{M}^T \mathbf{M}\,(\mathbf{x}-\mathbf{c})$$

The fact that the activity S(.) depends only on the distance (radius) $|\mathbf{x}-\mathbf{c}_i|$ from a center gave them the name *radial basis functions* RBF.

For adapting and scaling the ellipsoidal input field controlled by the matrix **M**, we used the scaling equation

$$\mathbf{M}^{NEW} = (\mathbf{I} - \gamma(1-\alpha)(\mathbf{a}\mathbf{a}^T))\,\mathbf{M}^{OLD} \qquad \mathbf{a} = (\mathbf{x}-\mathbf{c})/\,|\mathbf{x}-\mathbf{c}| \qquad (3)$$

with the scaling factor α and the learning rate γ, see (Pietruschka, Brause 1996).

There are principally two approaches to train the network parameters: either we train the two layers separately or as a whole.

The approach of treating the two layers separately, clustering the input space first and then optimizing the weights of the second layer, is fast, but it has some flaws. This gives us a high sample density of output values where we have clusters of input samples, not where the output error is high.

Therefore, we optimize both layers at the same time. To avoid the computational problems of the backpropagation (Rumelhart, Hinton, Williams 1986) approach we choose a different strategy. We start 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. This was already proposed for RBF nets, for example by (Schioler, Hartmann 1992). We insert the neuron at location $\mathbf{x}_k$, the k-th sample with the maximal error, that has to be compensated by the new m-th neuron. We have to design the width $\mathbf{M}_m$ such that it fits the new basis function in the context of all neighboring neuron basis functions. In contrast to the approach

of (Platt, 1992) we do not use gradient descend technique to rearrange all other neurons and adapt all their receptive fields: this is computionally intensive and is the source of new errors. Instead, we stop the adaption process of the new neuron by the criterion of non-significant activation on a data point. Additionally, we reduce the long distance neighborhood influence by a learning rate $\gamma(d)$ which drops with increasing distance from $\mathbf{c}_m$, that is with decreasing activity level, see (Pietruschka, Kinder 1995).

The whole growing and initialization learning algorithm, called GGRBF (growing generalized 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)
  AdaptTo1stNeighbor(M_m);        (* Eq.(3) *)
  level:=1; γ:=1;                 (* start with high activation level*)
  WHILE level> 0.01 DO
     FOR i:=1 TO |TrainingSet| DO
        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
```

4 Simulation results

The algorithm above was implemented and the approximation was simulated with real process data. An important key for the simulation performance turned out to be the two parameters: k, the number of neighborhood sampling points, and δ, the distance between the sampling points. The proper choice is determined by balancing the counteracting influences:

- If we choose δ too small, we increase the number of necessary sampling points for a certain neighborhood and increase therefore the dimension of the input space. Since we have only a small limited number of training samples, the training becomes very difficult since the input space becomes very sparse. On the other hand, if we choose δ too big, important information can be lost due to undersampling the dependency function,
- If we choose k too big, 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 too much context information, the generalization 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 a unnecessarily randomized training .

From the theoretical point of view, this is an interesting situation. Nevertheless, we are not yet aware of an applicable method of determining the optimal δ 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 δ and *k*) by an increment of 1 mm over the profile data of 5 profiles with the same values of G, E and P. This generated 1346 training patterns. The sixth profile was used for the generation of test set of 271 test points. Our multi-dimensional approximated function became

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

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 got different results. The best performance converged by training to the following expected absolute error, depending on the number of sampling points *k* and the interpoint distance δ. In Fig. 6, this is shown for k=7,9,11 for each of the intersample distances of δ=3,4,5 mm.

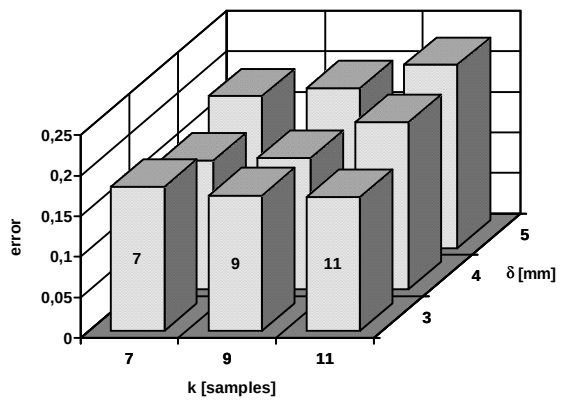

Fig. 6 The error development for different parameter values of δ and *k*

It is interesting to see that the error does not automatically decrease when we increase the number of sampling points. This is also valid if we consider instead the window size implied by the neighbor sample set size and the intersample distance δ, see figure 7.

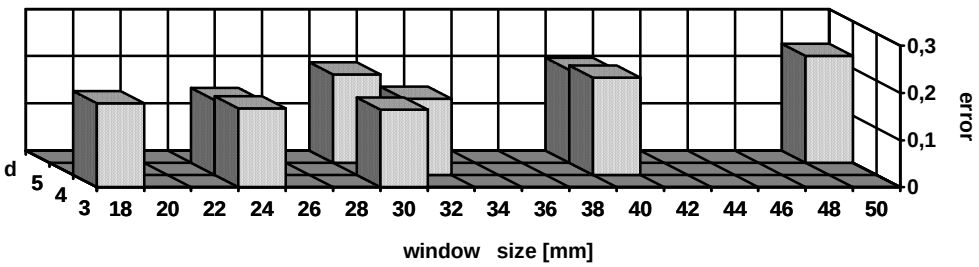

Fig. 7 The error development and the window size

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 δ=4 mm which corresponds to a window size of 32 mm. In Fig. 8 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.

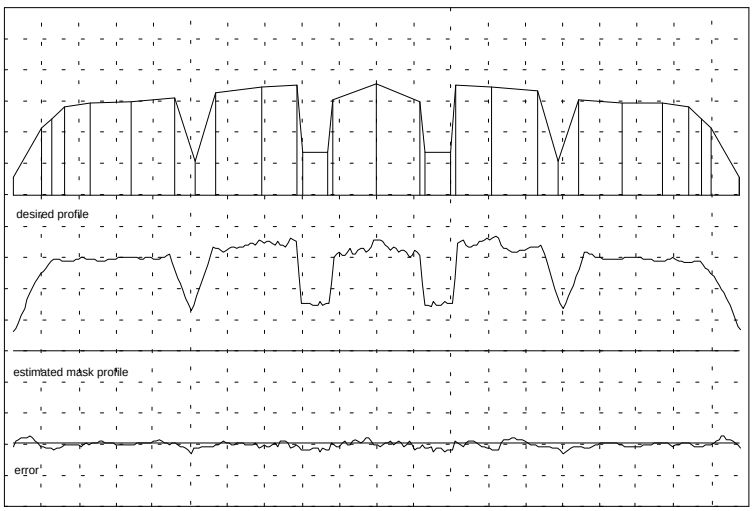

Fig. 8 The desired profile and the profile produced by the net for k=9, δ=4. For visualization purposes, the y-axis (the profiles and the error) scaling is

enlarged 3 times compared to the x-axis. One can clearly see that error is especially increased in the neighborhood of strong changes in the profile.

Now, why is there still such a big error? For example, let us regard the center of a profile. When we scale up the error, the drawing in Fig. 9 arises. The size of the sampling window is 32 mm whereas the width of the profile hill is 34 mm. Here, we can observe the typical influence of the sampling window: Since in the other training profile samples in the average there are no valleys (gaps) on the right and left hand side of the center outside the sampling window, the net „assumes“ the expansion pressure of more rubber from the right and left hand side. This would increase the profile at the center. But here, this is not the case, so without the anticipated rubber neighbor pressure the desired height in the center is not reached and an error occurs.

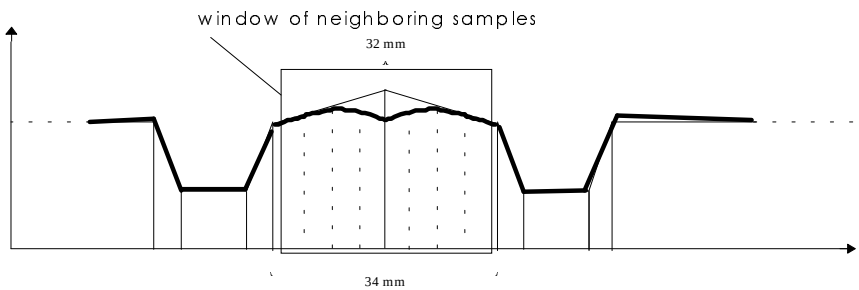

Fig. 9 The error and the sampling window size (k=9, δ=4). When the sampling window is too small, the deep ridges at the left and right hand side are not seen by the system and the prediction assumes too much rubber volume flow from the sides to the middle. This results in faulty estimation of the necessary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

In order to get rid of this effect, we have to enlarge the window and include the neighbor information about the desired gap, i.e. about the lack of rubber material in the neighborhood. If we do this, we have experienced: the error will go up also. Why?

This can be explained by the following. All samples r(j) near the point *i* of estimation in the profile are not completely independent of the value r(i): the profile does not consist of random

data. Thus, when we add more or less correlated samples we do not automatically add information to the training. Instead, the interdependencies ‘confuse’ the learning system. We have a sparse input space: if we enlarge the input space (by adding additional variables) without filling it with training patterns, the whole system learns less. This problem is known as the „curse of the dimensions“ (Huber 1985). The only remedy for this problem is either the augmentation of the number of training patterns (which is not available) or the appropriate deletion of unnecessary input lines, i.e. variables. The current neural methods for the latter concept include weight pruning (see e.g. Huberman, Rumelhart, Weigend 1991) or dimension reduction techniques by Kohonen maps (see e.g. Bruske, Sommer 1997). Nevertheless, to our knowledge there exist no analytical method yet for a proper selection of the distances and related numbers of the necessary sample points.

5 Discussion and outlook

In the previous sections we have presented an adaptive solution for the problem of unknown process parameters in tire 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 as the weight per meter and the resulting rubber profile. In Fig. 10 the layout of the adaptive control and the control migration from the human subject to the automatic system is shown.

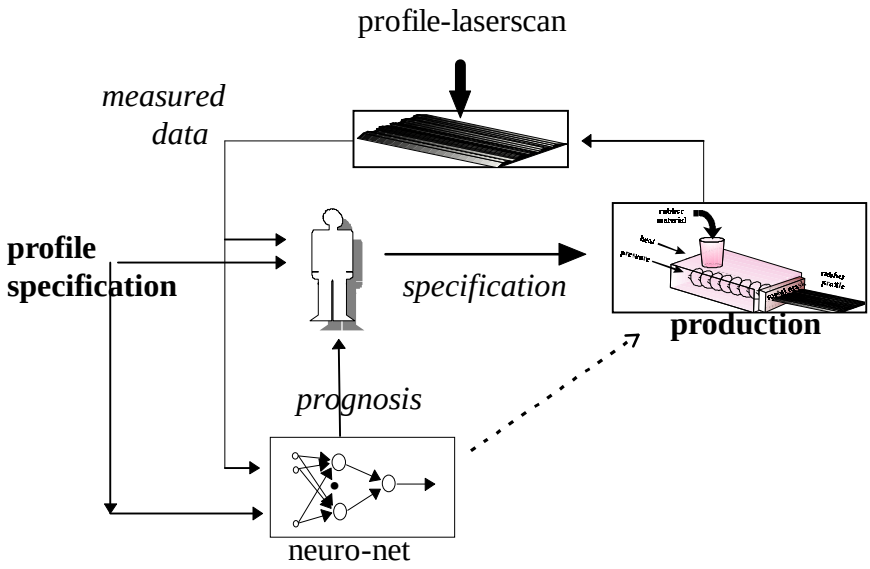

Fig. 10 The principal feedback lines in the profile specification. The specification is initially tied to the human operator who estimates the parameters for the conversion of the desired rubber profile to the metal mask profile. The produced rubber profile is then scanned and measured by a laser scan unit. This reveals an estimation error which is corrected by the human operator. With a neural network system, the estimation and correction process is automatic. According to the performance error and the obtained accuracy, the migration of the profile specification from the operator (solid arrow) to the neural system (dotted arrow) can be controlled by the operator himself.

In our case, the migration from the hand-made mask specification to the automatic, net generated version can be made gradually and smooth, leaving the control over the whole migration process always in the hand of the human

In the beginning the neural network profile prognosis is only used by the human operator as an additional information source. In Fig. 10 this is shown by a solid arrow. When the net has learned sufficiently and the prognostic error has decreased, the switch to the automatic profile generation (dotted line in Fig. 10) can be made. The human operator has always the full control over the usage of the adaptation process which is a strong source of human acceptance for all automation projects.

This approach has many technical advantages:

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

This results in solid economic advantages

- By involving a computer-based adaptive learning, the knowledge is automatically collected,
- The start for a tire production can be scheduled in a short time interval. This makes the production more flexible and competitive,
- Last not least, the minimization of the trial-and-error cycles saves a lot of money and resources,
- The better profile estimations save also rubber material. It is estimated that saving 1mm of rubber material on each profile will result in a save of about 0.8 Mill. $ per year. This is also interesting from the environmental point of view,
- The introduction of high-tech tires which contain multi-layer rubber profiles is a new challenge for the human designers. Here, the automatic adaptive estimation technique can help a lot to produce high quality tires with a short setup period only.

Also the human operator profits by the system:

- The task becomes less boring,
- The human operators are no longer tied exclusively to this production step.

Our work also shows that there are still several problems to be solved, for instance a method to overcome the "curse of dimensions" by a proper method to select independent input sample points for the network.

References

Buckley, J. (1992) Theory of the Fuzzy Controller: A Brief Survey; in: Cybernetics and Applied Systems, C. Negoita (Ed.), Marcel Dekker Inc., New York

Bruske, J., Sommer, G., (1997) Topology Representing Networks for Intrinsic Dimensionality Estimation; in W.Gerstner, A. Germond, M. Hasler, J. Nicoud (Eds.) Artificial Neural Networks-ICANN'97, Lecture Notes in Computer Science LNCS, Vol. 1327, Springer-Verlag Berlin Heidelberg New York, pp. 595-600

Haykin, S. (1994), Neural Networks, Maxwell Macmillan, Englewood Cliffs, NY

Hornik, K., Stinchcombe, M., White, H., (1989) Multilayer Feedforward Networks are Universal Approximators. Neural Networks, Vol 2, pp. 359-366, Pergamon Press

Huber, P.J. (1985) Projection Pursuit. The Annals of Statistics, Vol.13, No.2, pp.435-475 (with comments, pp. 475-525).

Huberman, B., Rumelhart, D., Weigend, A. (1991) Generalisation by Weight Elimination with Application to Forecasting, in: Lippmann, Moody (eds), Advances in Neural Information Processing Systems 3, Morgan Kauffmann Publ.

Platt, C. (1992) Learning by Combining Memorization and Gradient Descent, Proc. Int. Conf. Advances in Neural Information Processing Systems 4, Morgan Kauffmann Publ, pp. 714-720

Pietruschka, U., Kinder, M., (1995) Ellipsoidal Basis Functions for Higher-Dimensional Approximation Problems,. Proc. Int. Conf. on Artificial Neural Networks ICANN-95, Paris, EC2 & Cie, 92986 Paris La Défense, France, Vol. II, pp. 81-85.

Pietruschka, U., Brause, R. (1996) Using RBF Nets in Rubber Industry Process Control, in v.d.Malsburg, W. v. Seelen, J. Vorbrüggen, B. Sendhoff (Eds.): Artificial Neural Networks - ICANN 96, Lecture Notes in Computer Science LNCS, Vol.1112, Springer-Verlag, Berlin Heidelberg New York, pp. 605-610

D.E. Rumelhart, G.E. Hinton, R.J. Williams: Learning internal representations by error propagation; in [Rumelhard, McClelland 1986] Vol. I, pp.318-362

Rumelhart, D.E., McClelland, J., (1986) Parallel Distributed Processing. MIT press, Cambridge, MA

Schioler, H., Hartmann, U., (1992) Mapping Neural Network Derived from Parzen Window Estimator. Neural Networks, Vol.5, pp.903-909

Xu, L., Krzyzak, A., Yuille, A. (1994): On Radial Basis Function Nets and Kernel Regression: Statistical Consistency, Convergence Rates, and Receptive Field Size. Neural Networks, Vol. 7, No.4, pp.609-628

Narendra, K. (Ed.) (1986) Adaptive and Learning Systems, Plenum Press, New York

White, D. (Ed.) (1992) Handbook of intelligent control; neural, fuzzy, and adaptive approaches, Van Nostrand Reinhold, New York