Replacing FEA for sheet metal forming by surrogate modeling

Simon Wessing, Günter Rudolph, Sven Turck, Christian Klimmek, Sebastian Christopher Schäfer, Michael Schneider and Ulrich Lehmann

Cogent Engineering (2014), 1: 950853
Abstract: This paper presents an innovative approach for employing surrogate modeling in computer-aided design of sheet metal work pieces. The surrogate models replace expensive finite element analysis and thus shorten development time in the early design stages considerably. We experimentally compare neural networks and Kriging models for the task of predicting the sheet metal thickness after cold forming of B-pillars in automotive engineering. We also discuss some aspects of handling large data-sets and exploiting certain structures in the data.

1. Introduction
The automotive industry is one of the most important economic sectors worldwide (Womack, Jones, & Roos, 2007). In the past decades, many innovations like the Japanese concept of “lean production” which forced Western companies to re-invent themselves had been seen (West & Burnes, 2000). In this context, finite element analysis (FEA, Stasa, 1985) and computer-aided design (CAD, (Lee, 1999) emerged. The simulation of forming processes helps to reduce the development time and cost and thus increases the productivity (Ngaile & Altan, 2002). The current trend indicates the automotive industry’s demand for more complex forming process simulations but, simultaneously, shorter development times (Klimmek, 2010). Our aim is to replace the FEA for cold forming of sheet metal with a resource-saving surrogate model during the early stages of the virtual product development process. Based on a CAD geometry, the surrogate model is supposed to give shortcut
evaluations of some key figures, like sheet thickness or deformation, that determine within seconds if a part is technically feasible. Naturally, this task is quite ambitious and the predictions cannot be expected to be as precise as an FEA simulation. However, such a surrogate model could significantly shorten the development time of a new component and thus yields a huge economic benefit.

As the prediction has to be independent of the actual work piece, we have to extract some meaningful properties from the geometry and the forming process itself. To find such features is extremely challenging in the general case. Thus, we make the following relaxations to the task. First, the raw material must be given. This is not a severe restriction, as the material grades are standardized and we can train a separate model for each one. Additionally, we require that the training data have been collected from similar parts as the one that is to be evaluated. This should not be too unrealistic either, as companies usually spend a lot of work in a single area of operation and thus gain quite some data and experience. In total, we have 22 features that were finally extracted from the CAD model and the process. These features can be divided into three different subsets: material, process, and geometric parameters. The material parameters describe the formability of the given material. The process parameters, like sheet metal thickness, specify the actual forming process. The geometric parameters determine the surroundings of a specific point in the CAD geometry. Among others we use the features flatness, orientation, and symmetry (FOS) for the calculation of the geometric parameters. FOS is based on the flatness, orientation, convexity features proposed in Darbandi, Ito, and Little (2007), but the component convexity is replaced by symmetry. The material and process parameters actually used for the prediction are shown in Table 1. These are environmental parameters of the forming process, which means that they are constant within a work piece. Starting from the assumed default values, a latin hypercube sample of 32 FEA simulations is drawn in the ranges defined by column three of Table 1.

There exist some reports of related methods and applications in the literature. Nikitin, Nikitina, and Clees (2012) apply radial basis functions to FEA data of B-pillars, but only use process parameters and the original three-dimensional coordinates of points on the work piece to build the model. Thus, their model is not able to make predictions for different, unknown work pieces. Wagner, Paßmann, Weinert, Biermann, and Bledzki (2008) recognize the benefit of decomposing a response into different functional parts and modeling them separately, but do not investigate into any trends that may be given through known physical relations between input and output variables. In their application of a thermo-mechanical compaction process, the material temperature to be predicted strongly depends, among others, on a preheating temperature. However, they disregard this circumstance and use Kriging models with a constant mean. In Section 3, we analyze how much benefit the recognition of such special properties of an application may yield.

2. Methods

2.1. Preprocessing
The most critical points in sheet metal forming (SMF) lie in areas of high curvature, where the sheet thickness is in danger of falling below a critical minimum thickness. However, these areas only
amount to a small part of the surface. We have two options to tackle this problem of underrepresented regions: (1) either duplicate the important points, or (2) remove some of the other points. As the combined data-set of several FEA simulations may consist of hundreds of thousands points, option two is clearly the more favorable one. A prerequisite to both approaches is, however, to identify the critical points.

For Kriging models, several highly sophisticated approaches exist to efficiently fit models on large data-sets. Quiñonero-Candela, Rasmussen, and Williams (2007) give an overview of these techniques and compare them to the trivial approach of using only a subset of the data (SoD). However, as we are going to also evaluate a completely different type of surrogate model, we stick to SoD enriched by our expert knowledge.

To obtain the SoD, a deterministic reduction strategy was developed, which tries to identify and subsequently remove redundant data points. The similarity of two data points \( \vec{x}^{(1)} \) and \( \vec{x}^{(2)} \) is simply ascertained by calculating the Euclidean distance \( d(\vec{x}^{(1)}, \vec{x}^{(2)}) = \left( \sum_{i=1}^{m} (x_i^{(1)} - x_i^{(2)})^2 \right)^{1/2} \) in the \( m=22 \) input dimensions. Next, an \( m \)-dimensional \( k \)-d tree is created into which the data points are sorted using the calculated distances \( d \) to each other. Sorting the data according to their distance to each other has the advantage that extremely similar and thus redundant data points will end up in the same part of the \( k \)-d tree as leaves of the same parent. This way, it is possible to prune the tree from the bottom up. During this process, all but one leaf of a parent are removed whose nearest-neighbor distance lies below a certain minimum distance threshold. This way, we can ensure that the remaining data are equally distributed over and representative of all dimensions \( m \). To calculate the minimum distance threshold, first of all, distances between nearest neighbors are sorted, such that \( d_1 \leq d_2 \leq \ldots \leq d_n \). The actual minimum distance threshold \( \tau \) is then determined by

\[
\tau = \frac{1}{\sqrt{2}} \sum_{i=[\tau]}^{n} d_i
\]

The resulting data-set represents an equally distributed subset of the original data-set without the redundant data. See Figure 1 for a comparison of the histogram for the complete set and the reduced data-set.

Figure 1. The histogram of the distances for all data points for the original (blue) and reduced (green) data-set.
For the problem at hand, the previously described method is augmented with three heuristic rules describing points that are to be preserved in the data-set. The first rule says to keep the points corresponding to $2 \times m$ extreme values (minimal and maximal values of each dimension) of the original data-set. In a second step, we take a look at the difference $z = y - x^2$ between the initial sheet thickness $x^2$ and the resulting sheet thickness $y$ as predicted by the FEA. We preserve all points $\vec{x}_i$ for which $y_i \leq \frac{1}{3} z_{\text{min}}$ or $y_i \geq \frac{1}{3} z_{\text{max}}$. An exploratory analysis revealed that the distribution of $z$ has very thin tails in our application, so that only a small percentage of points fulfill this criterion. The extreme input and difference values are assumed to define outliers that contain valuable information about the forming process. Finally, all data points in regions of high curvature are preserved, too.

To sum up, we would like to say that although the current feature extraction and data reduction procedure is probably not yet a final solution, it is a first step towards a generic scheme for surrogate modeling of SMF processes. For the time being, we will try to compensate any shortcomings with specially tailored surrogate models, namely a multilayer perceptron with customized training algorithm and a Kriging model for large data-sets.

2.2. Artificial neural networks

In surrogate modeling, the feedforward multilayer perceptron model in conjunction with the backpropagation training algorithm as a method of gradient descent learning is widely used (Rojas, 1996). The backpropagation training algorithm is suitable for quick and reliable learning. Yet, the training result is not predictable as it depends on the random initialization as well as the architecture of the artificial neural networks (ANN). Thus, a large number of ANN with varying initializations as well as different architectures, i.e. number of hidden layers (HL), neurons per HL, and activation functions, need to be trained to achieve desired results. As finding the appropriate combination of initialization and architecture by enumeration is computationally too expensive, we end up with a mixed, non-linear optimization problem, which can only be solved approximately. To do this, one could, for example, simply do a random search. More sophisticated optimization algorithms that can cope with mixed decision variables are, for example, evolutionary algorithms (EA) (Bäck, 1996), which are employed here. In particular, a two-stage training procedure was developed and implemented. In an outer optimization loop, an EA searches for the optimal ANN architecture for the given data-set. In the inner loop, this architecture is used in the gradient descent training of multiple ANN with different initializations using a random forest-inspired method, dubbed “training with randomly selected data” (TRSD), to determine the optimal model for the given data-set.

2.2.1. Evolutionary architecture optimization

The developed evolutionary architecture optimization method is a modified version of a standard genetic algorithm as described by Bäck (1996). Figure 2 shows an activity diagram detailing the algorithm’s structure. At the start of the optimization process, a parent population of $\mu$ ANN individuals is initialized. The search space consists of the number of HL, the number of neurons per HL, as well as the activation function of the HL. During initialization, these values are randomly set for each of the $\mu$ individuals within user-specified bounds (e.g. of these bounds, see Table 2). After the random generation of ANN architectures, a short initial training is carried out for each of them. This training comprises the creation of several instances for each architecture with different initializations of bias and weight values and subsequently a gradient descent training for a small number of epochs. The best performing bias and weight initialization of each architecture is then selected for use in the evolutionary optimization process, while the rest are discarded.

The optimization loop starts with the backpropagation training of all $\mu$ parents for a user-set number of training epochs. In every iteration, this training is resumed, albeit only for half the number of epochs used in the previous iteration. This means that a new evaluation for each individual has to be done in every iteration. Next, the parent individuals are selected in a multi-round tournament to create offspring through uniform crossover, which means that for every gene of the child, it is randomly decided which parent will provide the values of the gene. As the number of genes is dynamic due to the variable number of HL, it may happen that the parent selected by the crossover function has
fewer HL than the child. Thus, the number of neurons for this HL is missing and will therefore be randomly initialized using the same bounds as before. The last step of the recombination process is the initialization of the child’s bias and weight values by recombining the parents’ values. During this process, the weight and bias values of the parent whose gene was used for the initialization of the child’s respective HL are copied to the child. The recombination process is repeated until a population of $\lambda$ children is created. Of these individuals, a user-set percentage has one of its genes randomly mutated. As the genes of each individual can be divided into three categories, the mutation function behaves slightly different for each type. The three possibilities are changing the number of

<table>
<thead>
<tr>
<th>EA parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>#Generations</td>
<td>7</td>
</tr>
<tr>
<td>#Parents</td>
<td>50</td>
</tr>
<tr>
<td>#Children</td>
<td>30</td>
</tr>
<tr>
<td>#Elite parents</td>
<td>20</td>
</tr>
<tr>
<td>#Tournament rounds</td>
<td>7</td>
</tr>
<tr>
<td>#Children mutated</td>
<td>16.6%</td>
</tr>
<tr>
<td>#Hidden layers</td>
<td>1 \ldots 3</td>
</tr>
<tr>
<td>Size of each HL</td>
<td>20 \ldots 120</td>
</tr>
<tr>
<td>HL activation function</td>
<td>Logistic $P(t) = \frac{1}{1 + e^{-t}}$</td>
</tr>
<tr>
<td>#Training epochs</td>
<td>20 \ldots 60</td>
</tr>
<tr>
<td>Learning rate</td>
<td>0.1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TRSD parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>#Data per training</td>
<td>15%</td>
</tr>
<tr>
<td>#Unique ANN initializations</td>
<td>22</td>
</tr>
<tr>
<td>#Consecutive trainings per init</td>
<td>24</td>
</tr>
<tr>
<td>#Epochs per training segment</td>
<td>10</td>
</tr>
<tr>
<td>Learning rate</td>
<td>0.1</td>
</tr>
</tbody>
</table>
HL, modifying the number of neurons on a HL, and randomly exchanging the currently selected activation function with another one. If the number of HL is reduced, one or more of the existing HL are dropped beginning at the end. If, on the other hand, one or more HL are added, they are randomly initialized in the same way as before. If the number of neurons per HL is to be mutated, the mutation function behaves similarly by either removing neurons from the particular HL or adding new randomly initialized ones. After the mutation process, the population is trained by backpropagation and subsequently performance evaluated. The next generation parent population of $\mu$ individuals is selected by picking the elite best parent from the current generation and the $\lambda$, elite best-performing children. This whole loop is repeated for a number of generations after which the final population of parent individuals is evaluated using a set of test data. The architecture of the best-performing individual is then selected for a final training stage.

2.2.2. Training with randomly selected data

The “TRSD” is inspired by the random forest method (Breiman, 2001), which is well known in the machine learning community. As the amount of data used during training has a big influence on the training outcome of ANN, we developed a method that splits up the training process of an ANN into smaller chunks with a limited number of training epochs each. Thus, instead of training an ANN with the full data-set for a larger amount of epochs, we select a subset of the training data purely at random and train the ANN with it only for a few epochs. The resulting trained ANN is then continued to be trained in the next chunk with another randomly chosen subset of the full data-set. This process is executed for a user-set number of different ANN initializations in parallel, resulting in a large number of trained ANN, each of which only requires a fraction of the time it would take to train with the full data-set. The best-performing ANN is determined by evaluating each ANN resulting from every chunk of training and every initialization with the set of test data.

2.3. Kriging models

Kriging is a modeling approach, which was originally developed in geo-statistics. Nowadays, it is also widely used in engineering (Forrester, Sbester, & Keane, 2008). Its attractiveness stems from its reliable behavior, e.g. guaranteeing to exactly interpolate the given training data and producing a continuously differentiable surface. These guarantees, however, come at the cost of more rigid assumptions about the process to be modeled. Ordinary Kriging, for example, assumes that the process to be modeled is stationary, which means that it has a constant mean over the whole domain (Santner, Williams, & Notz, 2003). However, this property is not given here, because the predicted sheet thickness $y$ is highly dependent on the input sheet thickness $x_2$ (see Figure 3). On the B-pillar data-set, a highly significant correlation of .82 was observed. Thus, it would be advisable to only let the Kriging model predict the residual $y - x_2$. This approach is also mentioned in Rasmussen and Williams (2006, p. 27) as Kriging with a fixed mean function. It does not require any changes to the Kriging implementation, as the necessary subtractions and additions can be carried out before and
after the data are processed by the surrogate model. The benefit of this approach is experimentally verified in Section 3.

The exponential correlation function \( c(x_1, x_2) = \exp\left(-\sum_{i=1}^{m} \theta_i |x_i^{(1)} - x_i^{(2)}|\right) \) is used as a kernel in this paper. The \( \theta_i \) in this formula, so-called activity parameters, have to be adjusted for a good model fit. We are doing this by optimizing the 10-fold cross-validation error. Preliminary experiments indicated that exponential correlation produces better and faster models than the Gaussian correlation function on this data-set. This is plausible, because through the abstraction of the original coordinates (see Section 1), it is not guaranteed that the resulting response surface is smooth or even continuous. The correlation function \( c \) is used to compute a correlation matrix \( C \). Building the model requires to compute the inverse \( C^{-1} \), which generally takes \( O(n^3) \) operations. Thus, Kriging becomes computationally infeasible for data-sets of \( n \geq 100 \) points. Consequently, we developed a heuristic to partition the data-set and to fit a Kriging model of feasible size on each partition. The rationale behind this heuristic is as follows: due to the way of how, for example, flatness and orientation measures are computed, some of the input dimensions contain truncated distributions (see Figure 3). This means that the points corresponding to these truncated values lie in a \( m-1 \) dimensional subspace and can be separated from the rest by a paraxial hyperplane. Our heuristic rule now says to split the data-set in the dimension with the highest amount of truncated values, under the side constraint that at least 2.5\% of the data can be separated. By applying this rule recursively, a binary tree containing a Kriging model in each leaf can be built. This way, we can achieve a partition of the data-set and a dimension reduction in some partitions at the same time. We are restricting the tree depth to 3, meaning that at most \( 2^3 = 8 \) models can be built. If any subset contains more than 1,000 points and cannot be partitioned further, we take a random sample of 1,000 points to fit the model.

3. Experiment

3.1. Research question

Our experiment focuses on three decision areas practitioners often have to deal with in metamodeling. These are model selection, preprocessing of data, and incorporation of expert knowledge. For each area, two options are chosen to be compared in a full-factorial experiment. Our research question then is to find out how big the influence of a decision on a model type, a data reduction mechanism, and the trick of modeling the residuals is.

3.2. Pre-experimental planning

Preliminary experiments showed that \([10^{-6}, 10^0]^m\) is an advisable search space for the \( \theta \) parameters of Kriging. Although the lower bound seems to be very small, it is probably useful for the high-dimensional input space, because low values of \( \theta \) allow the model to assign a very low influence to a dimension \( i \). It is an interesting observation that the approach to model the thickness residuals is also applicable to the ANN. However, the question arises if an ANN can benefit from it in the same way, as ANNs do not contain any assumption of stationarity of the modeled function.

3.3. Task

To measure the model quality, we are using the 97.5\% quantile of the squared prediction errors over some test set of data. The rationale behind choosing this measure is that we assume it more important to obtain reliable results than to minimize the mean squared error. For each configuration of model, data reduction, and predicted variable, the task is to obtain an error as small as possible. The results are compared to a conventional multilayer perceptron with backpropagation learning as a reference model.

3.4. Setup

The parameters used for the EA and TRSD optimization processes of ANNs are detailed in Table 2. As the training algorithms for both models and the random data reduction are stochastic, we make 15
repeats for each configuration. The training set for fitting the models consists of four B-pillars of existing car bodies. These examples were provided by ThyssenKrupp Steel. The FEA was done with LS-DYNA Version Is971s R4.2.1 of Livermore Software Technology Corporation. The steel grade simulated in the FEA is DP-W 600, a hot rolled dual phase steel. As we are interested in virtual product development, the FEA data are the “true” reference values for us.

From the FEA data of the parts, we obtained $4 \times 10^6$ points in total, which were reduced to $5 \times 10^5$ points via the random and the custom data reduction method, respectively. In case of the custom reduction method, $1.16 \times 10^5$ points were chosen through the three heuristic rules, while the rest was determined by the method described in Section 2.1. The test set is one B-pillar (see Figure 4) that was designed independently, but subjectively looks similar to the ones in the training set from a human’s point of view. This test set contains $7.5 \times 10^5$ points. This setup resembles the planned use case of aiding the construction of new variants of existing parts.

3.5. Results
The results of the two models for the different settings are shown in Figure 5. We can see that the performance of the reference model is quite bad. All of the new methods can improve these results significantly. While there seems to be little influence for the data reduction and prediction variable on the neural network, Kriging clearly works best with the custom data reduction and predicting the deviation $y - x$. Kriging also beats the neural networks by a large margin.
3.6. Discussion
The experiment shows that the modeling precision can be substantially improved by carefully incorporating expert knowledge into the process. However, the employed model still has the largest influence on the performance. While neural networks may have their fields of application, this does not seem to be the right one. Kriging models not only achieve a smaller median error over the 15 repeats (see Figure 5), but also a much smaller variance in the results, which means that the training algorithm is much more reliable.

4. Conclusions and outlook
In this work, we showed the superiority of Kriging models over neural networks in a prediction task for SMF. Additionally, it proved beneficial to carry out a data reduction by finding a uniformly distributed subsample of the originally large data-set. Furthermore, it is indeed worthwhile to conduct an explorative data analysis to find exploitable correlations.

A real-world application of the most promising model variant will have to show if the currently obtained precision is already sufficient to aid engineers in designing new parts. Future efforts to improve the accuracy will probably have to focus on the data generation. A primary objective should be to reduce the number of necessary input dimensions \( m \). For this purpose, a statistically driven dimension reduction, e.g. through a principal component analysis, could be an easy way to lower execution time and memory consumption of the modeling process, while maintaining or improving prediction accuracy. However, this would prevent the application of our trick concerning the sheet thickness deviation and the partitioning strategy with dimension reduction. Otherwise, the preprocessing of the data, where abstract geometric features are extracted from the FEA data, would have to be revised.

Acknowledgements
The authors thank Christian Radisch and Marcel Stieglitz for contributions to the used software.

Funding
The work in this paper is based on an interdisciplinary research project of the three partner institutions Technische Universität Dortmund, South Westphalia University of Applied Sciences, and SIMUFORM GmbH. The project ran from September 2010 to February 2012 and was funded through the German Federal Ministry of Economics and Technology as part of the Central Innovation Programme SME (ZIM) [grant number KF244203WDD].

Author details
Simon Wessing¹
E-mail: simon.wessing@tu-dortmund.de
Günter Rudolph¹
E-mail: guenter.rudolph@tu-dortmund.de
Sven Turck²
E-mail: turck@simuform.de
Christian Klimmek²
E-mail: klimmek@simuform.de
Sebastian Christopher Schäfer¹
E-mail: scoefer.sebastian@fh-swf.de
Michael Schneider¹
E-mail: schneider.michael@fh-swf.de
Ulrich Lehmann⁳
E-mail: lehmann.ulrich@fh-swf.de
¹ Fakultät für Informatik, Technische Universität Dortmund, 44227 Dortmund, Germany.
² SimuForm GmbH, 45699 Herten, Germany.
³ Institute for Computer Science, Vision, and Computational Intelligence, Fachhochschule Südwestfalen, 58644 Iserlohn, Germany.

Citation information

Note

References
Nikitin, I., Nikitin, L., & Clee, T. (2012). Nonlinear metamodeling of bulky data and applications in automotive design. In M. Günther, A. Bartel, M. Brunk,


S. Schöps, & M. Striebel (Eds.), Progress in industrial mathematics at ECMI 2010, mathematics in industry (Vol. 17, pp. 295–301). Berlin: Springer.


© 2014 The Author(s). This open access article is distributed under a Creative Commons Attribution (CC-BY) 3.0 license.

You are free to:
Share — copy and redistribute the material in any medium or format
Adapt — remix, transform, and build upon the material for any purpose, even commercially.
The licensor cannot revoke these freedoms as long as you follow the license terms.

Under the following terms:
Attribution — You must give appropriate credit, provide a link to the license, and indicate if changes were made.
You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.
No additional restrictions
You may not apply legal terms or technological measures that legally restrict others from doing anything the license permits.

Cogent Engineering (ISSN: 2331-1916) is published by Cogent OA, part of Taylor & Francis Group.
Publishing with Cogent OA ensures:
• Immediate, universal access to your article on publication
• High visibility and discoverability via the Cogent OA website as well as Taylor & Francis Online
• Download and citation statistics for your article
• Rapid online publication
• Input from, and dialog with, expert editors and editorial boards
• Retention of full copyright of your article
• Guaranteed legacy preservation of your article
• Discounts and waivers for authors in developing regions
Submit your manuscript to a Cogent OA journal at www.CogentOA.com