

University of Duisburg-Essen
Faculty of Computer Engineering

Bachelor Thesis

**Mathematical Methods to Ascertain Flow-Curve
Functions With the Help of Neural Networks**

by Mihail Mihaylov

SS 2006

Matriculation number: DS 02104390 01

1st supervisor: Prof. Dr. rer. nat. Johannes Gottschling
Institute for Applied Material Science

2nd supervisor: Prof. Dr. -Ing. P. J. Mauk
Institute for Applied Material Science

RESERVED for description of thesis

Acknowledgement

I would like to thank Prof. Dr. rer. nat. Johannes Gottschling, Prof. Dr. -Ing. P. J. Mauk and Dr. -Ing. Heiko Gemming for their excellent supervision and extensive support throughout my work on this thesis. Their continuous concern and user-friendly attitude is more than exemplary.

I would also like to thank my mother for the opportunity she gave me to study and graduate in the University of Duisburg-Essen. Without her **support**, this thesis would not exist.

I hereby formally declare that the work submitted is entirely my own and does not involve any additional human assistance. All help materials used have been cited appropriately with the corresponding bibliographical references provided to the best of my knowledge.

December 2006

Contents

Abstract.....6

1. Introduction to Flow Curves of Metallic Materials.....7

 1.1. Parameters of Flow Curves.....7

 1.2. Cold Flow Curves.....8

 1.3. Hot Flow Curves.....9

2. Regression Analysis.....11

 2.1. Linear Regression Analysis.....11

 2.2. Nonlinear Regression Analysis.....12

 2.3. Coefficient of Determination.....13

3. Regression Analysis for Flow Curves in Metal Forming.....13

4. Artificial Neural Networks.....14

 4.1. Functionality of Artificial Neural Networks.....15

 4.2. Over-Learning of Artificial Neural Networks.....15

 4.3. Performance of Artificial Neural Networks.....16

5. Artificial Neural Networks in Metal Forming.....16

6. Combination of Methods.....19

7. Realization of the Combined Model.....19

 7.1. Training a Neural Network.....20

 7.1.1. SNNS Converter.....20

 7.1.2. Stuttgart Neural Network Simulator.....21

 7.2. Creating a Recall Function from the Trained Network.....23

 7.3. Calling the DLL from within software.....25

 7.3.1. “Flow Stress Plotter”.....26

 7.3.2. Usability of “Flow Stress Plotter”.....27

8. Conclusion.....28

References.....29

Appendix I (UML Sequence Diagram, Explanation of Elements).....I-1

Appendix II (Producing a DLL from Training Data).....II-1

Appendix III (Flow Stress Plotter, User’s Manual).....III-1

Appendix IV (Flow Stress Plotter, File Conventions for Compatibility).....IV-1

Abstract

Computations of cold and hot flow curves have long been used in metal forming via isolated mathematical functions, that make the calculation of smooth and precise material flow curves extremely difficult for wide temperature ranges. When modern statistical methods are integrated into a computer environment in the form of software application, accurate flow curves for different steel grades can be approximated. These methods, however, have their advantages and disadvantages, but when combined in a proper way, certain model arises, which puts together the benefits and eliminates most of the drawbacks of these techniques.

This document explores the application of regression analysis and artificial neural networks in metal forming in order one to obtain values of flow stress for the range 20° – 1200° C. The computation of flow stress is carried out by software – “Flow Stress Plotter”, developed by the author, having also the purpose to visualize flow curves in two as well as in three dimensions.

1. Introduction to Flow Curves of Metallic Materials

For each computation in metal forming, the knowledge of material flow curves is a fundamental requirement. The flow curve of a material is the graph of a function, which indicates the flow stress k_f (Kf) as a function of the strain φ (Phi), strain rate $\dot{\varphi}$ (Phip) or temperature T (Temp). Flow curves have several characteristic shapes. For example one of them is “blue brittleness”, which is the reduced ductility occurring as a result of strain aging, when certain ferrous alloys are worked between 150° and 400° C [1].

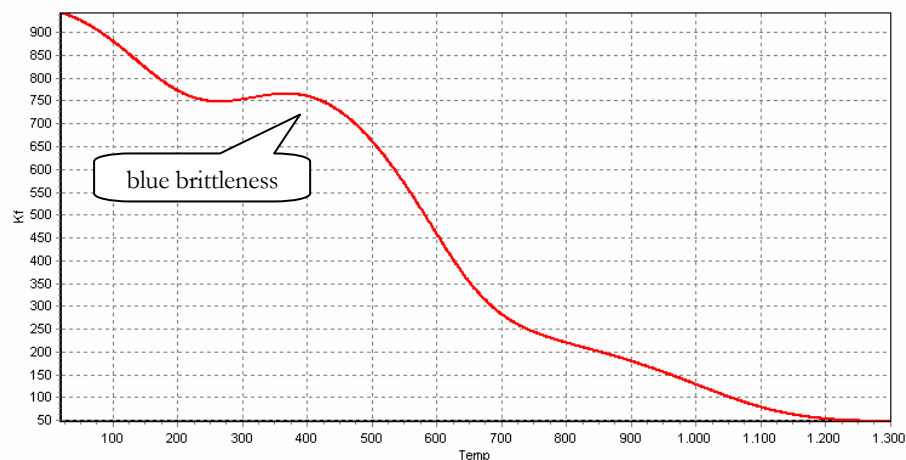


Fig. 1 – Flow curve of material 100Cr6 with $\Phi=0.5$ and $\dot{\Phi}=1.5 \text{ s}^{-1}$

1.1. Parameters of Flow Curves

In the theory of metal forming certain relations exist between the metal forming parameters strain (φ , [-]), strain rate ($\dot{\varphi}$, [s^{-1}]), temperature (T , [$^{\circ}\text{C}$]) and flow stress (k_f , [$\text{N} \cdot \text{mm}^{-2}$]). The parameter strain (φ , [-]) is defined as the deformation produced on a body by an outside force [1] and is computed by the formula (valid for cylindrical samples):

$$\varphi = \ln \left(\frac{h_0}{h_1} \right) \quad (1)$$

where h_0 is the initial height and h_1 is the resulting height after compression.

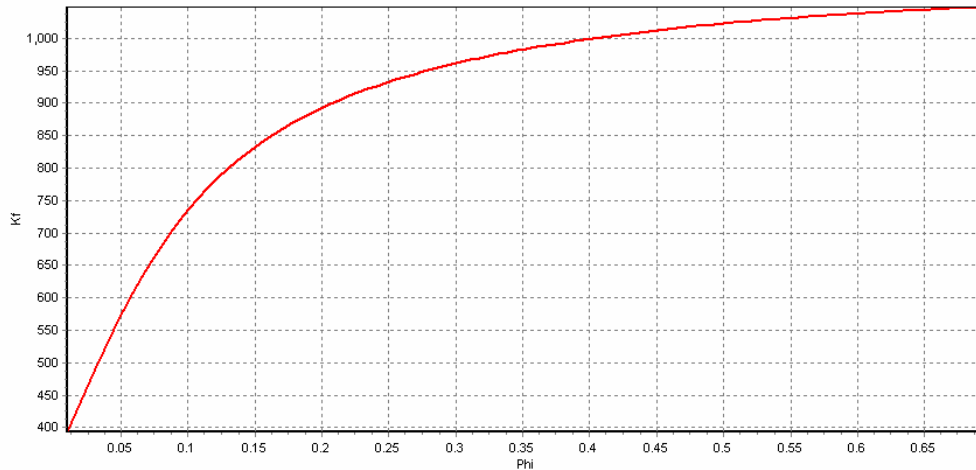


Fig. 2 – Flow curve of material 100Cr6 with $T=20^{\circ}C$ and $Pbip=1.5 s^{-1}$

The strain rate ($\dot{\varphi}$, [s^{-1}]) is the change in strain over time and is defined as:

$$\dot{\varphi} = \frac{\Delta\varphi}{\Delta t} \quad (2)$$

The flow stress (k_f , [$N \cdot mm^{-2}$]) is the externally applied shear stress (σ , [$N \cdot mm^{-2}$]), which is required to cause plastic deformation of solid metals [1].

1.2. Cold Flow Curves

A well-known model function for cold flow curves for fixed temperature and strain rate is the Ludwik equation:

$$k_f = A \cdot \varphi^B \quad (3)$$

which offers a good approximation of the flow stress, but is not exact for very small and very big strains. The Institute for Applied Material Science (IAM) has searched many years for new model functions for cold flow curves that give better approximation than the Ludwik equation. There is no universal model function, that can approximate the flow stress for all steel grades. There are, however, functions that show good results only for certain groups of steel grades. Some of them are:

$$k_f(\varphi) = A \cdot e^{B \cdot \varphi} - \frac{1}{(\varphi + E)^F} + G \cdot \varphi + K \quad (4)$$

$$k_f(\varphi) = A \cdot \varphi^C - \frac{1}{(\varphi + E)^F} + G \cdot \varphi + K \quad (5)$$

and

$$k_f = A + e^{\frac{(\ln \varphi - B)^2}{C}} \quad (6)$$

where A, B, ..., K are material dependent constants.

1.3. Hot Flow Curves

Empirical evidences have shown that for hot flow curves the relations between the above parameters are as follows:

- Flow stress – strain relation:

$$k_f \sim \varphi^{(M_3 + f_1(T, \varphi))} \quad (7)$$

- Flow stress – strain rate relation:

$$k_f \sim \dot{\varphi}^{(M_2 + f_2(T, \varphi))} \quad (8)$$

- Flow stress – temperature relation without transformation processes:

$$k_f \sim e^{M_1 \cdot T} \quad (9)$$

where f_1 and f_2 are some special given functions.

Furthermore, other parameters influence metal forming in this temperature range, such as the dynamic softening, caused by recovery and re-crystallization processes. Its relation to the flow stress can be described as:

$$k_f \sim e^{M_4 \cdot \varphi} \quad (10)$$

These relations have lead to some equations like:

$$k_f = M_0 \cdot e^{M_1 \cdot T} \cdot \dot{\varphi}^{M_2} \cdot \varphi^{M_3} \cdot e^{M_4 \cdot \varphi} \quad (11)$$

$$k_f = M_0 \cdot e^{M_1 \cdot T} \cdot \dot{\varphi}^{M_2} \cdot \varphi^{M_3} \cdot e^{\frac{M_4}{\varphi}} \quad (12)$$

and

$$k_f = M_0 \cdot e^{M_1 \cdot T} \cdot \dot{\varphi}^{M_2 + M_5 \cdot T} \cdot \varphi^{M_3} \cdot e^{M_4 \cdot \varphi} \quad (13)$$

where M_0 through M_5 , as in the cold flow curves, are coefficients that are determined by the 21 analysed alloy concentrations (C, Si, Mn, Cr, Ni, W, Co, Mo, V, Ti, Al, Nb, Cu, B, N, Zr, Sn, Ce, Pb, P, S).

Material	C	Si	Mn	P	S	Cu	Cr	Al	Ni	N	B	Ti	Mo	PB	V
19MnB4Cr	0,2210	0,0260	0,8400	0,0150	0,0060	0,0120	0,2940	0,0440	0,0260	0,0045	0,0029	0,0380	0,0040		
X3CrNiCu	0,0110	0,3500	0,8400	0,0220	0,0060	3,2900	17,6000		9,7400	0,0270			0,1700		0,0900

Fig. 3 – Table of the alloy concentrations of two different steel grades

The Institute for Applied Material Science has found that equation (13) is good for extrapolation, when used as a model function.

One important disadvantage of isolated cold and hot flow curves is the limited possibility to cover wider ranges from lower to higher temperatures [2] and to fit the values in the boundary region between both curves.

In order one to obtain a smooth flow curve, that represents the physical behaviour of certain material from room temperature (RT) to approximately 1200° C, one has to carry out measurements, extremely dense within the specified range, which is practically not feasible. For this reason measurements are taken only from a few points of the range and then interpolation and extrapolation is used to cover the whole desired problem space. Several methods exist to perform the latter, but only some of them show realistic results.

2. Regression Analysis

One method, which involves interpolation and extrapolation, is regression analysis. It uses mathematical techniques to find the parameters of a model function, so that it can best represent and predict the behaviour of a data set. Figure 4 shows the use of a model function.

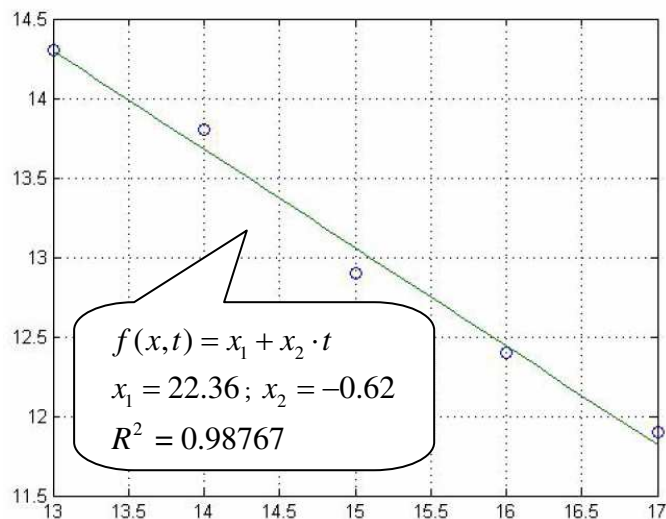


Fig. 4 – Graph of model function $f(x,t) = x_1 + x_2 \cdot t$

If the relationship between two random variables is to be modeled, one speaks about simple regression, whereas multiple regression shows the dependency between three or more variables.

2.1. Linear Regression Analysis

Linear regression is a statistical method of deriving functional relation between measured values. The term “linear” means that a model function, linear in parameters, rather than in variables, is used to relate the response variables to the explanatory variables (or predictors). Such a model function can be stated in the form:

$$y = F(t, x) = x_1 f_1(t) + x_2 f_2(t) + \dots + x_n f_n(t) \quad (14)$$

where $f_i(t)=1$; $x=(x_1, x_2, \dots, x_n)$ is a vector of n parameters (or regression coefficients); $t=(t_1, t_2, \dots, t_m)$ is a vector of m predictor variables; and $F(t, x)$ is the model function that relates t to the measured data, using the parameter vector x . Response variables are those, whose

values vary when the values of the independent ones are changed. Alternatively, predictors are the variables, which are manipulated in order to determine their relationship to the dependent ones. The regression coefficients x_i are computed by the method of least squares that minimizes the sum of squares of the residuals between points generated by the function and corresponding points in the data [3].

2.2. Nonlinear Regression Analysis

The idea behind nonlinear regression is the same as that of linear regression, namely to relate a response variable y to a vector of predictor variables t . Nonlinear regression uses model functions, that are not linear in parameters, rather than in variables, as in

$$y = f(x, t) = x_1 \cdot e^{x_2 \cdot t_1 + x_3 \cdot t_2} \quad (15)$$

for example, where the model function $y = f(x, t)$ is nonlinear in the parameter vector x .

Several methods exist for solving nonlinear equations. Two of them are the Gauss-Newton method and the Levenberg-Marquardt method, but they are difficult to carry out by hand. For this purpose, certain software solutions exist (e.g. MATLAB) that can solve such kind of equations. When a certain problem requires a nonlinear model, investigators usually try to state the problem in a specific way, so that the relationships between variables can be transformed into linear ones. For example, the well-known Ludwik equation for cold flow curves

$$k_f = A \cdot \varphi^B \quad (16)$$

can be transformed into a linear relationship in the form

$$\ln y = \ln A + B \cdot \ln \varphi \quad (17)$$

which can also be written as $\tilde{y} = \tilde{A} + B \cdot \tilde{\varphi}$, since the computation of linear regressions is much simpler and their statistical properties are better known.

2.3. Coefficient of Determination

The quality of approximation for a model function can be rated by the regression factor R^2 (R-squared). This is called coefficient of determination and is a value between 0 and 1, that shows the relative predictive power of a model or, in other words, the ratio of the estimated dispersion to the total dispersion around the mean measured value \bar{y} [6]. The formula for \bar{y} is

$$\bar{y} = \frac{1}{m} \cdot \sum_{i=1}^m y_i \quad (18.1)$$

where m is the number of measured values y_i .

The closer R^2 is to 1, the greater ability to predict the model has; and respectively – a value close to zero suggests a poor model. The coefficient of determination, however, do not tell whether an included independent variable is statistically significant, or whether the measured values can be well interpolated or extrapolated with the help of the model function. The formula for the computation of R^2 is

$$R^2 = \frac{\sum_{i=1}^m (F(t_i, x) - \bar{y})^2}{\sum_{i=1}^m (y_i - \bar{y})^2} \quad (18.2)$$

3. Regression Analysis for Flow Curves in Metal Forming

If one is to model the behaviour of a material in a flow stress diagram from RT to approximately 1200° C, he or she needs to have some empirically measured data, upon which a good model function can be build and using regression analysis the coefficients can be determined, in order one to obtain a smooth flow curve. Figure 5 shows one of the best model functions developed at IAM that covers the complete range from RT to 1200° C.

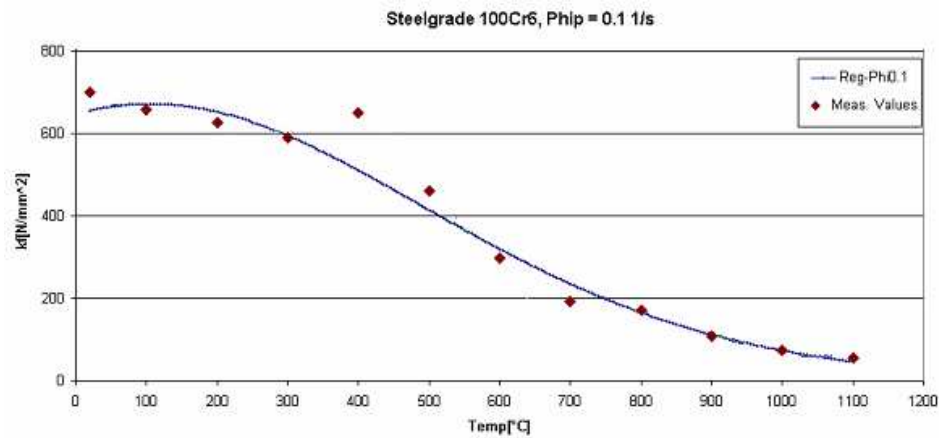


Fig. 5 – Model function for the range $20^{\circ} - 1100^{\circ}\text{C}$ for material 100Cr6 with $\Phi_i=0.1$ and $\Phi_{ip}=0.1 \text{ s}^{-1}$

As it can be seen from the graph, the analysis shows quite good results when extrapolating the values (above 800°C , for which the function is suitable). Although the model function has a relatively high regression coefficient ($R^2 = 0.976388$), it does not represent exactly the physical behaviour of the material and is not suitable for interpolation. It can also be proved that good function types for hot flow curves are not appropriate for temperatures below 800°C .

Another disadvantage of regression analysis is that it needs a defined function type for determining the regression coefficients, based on a data collection as input. Such a unique function type, that covers the range from room temperature (RT) to high temperatures, is up to now not known [4].

4. Artificial Neural Networks

A different approach to finding a unique function to describe cold as well as hot flow curves, that utilizes interpolation and extrapolation, is the use of artificial neural networks. They are designed to model some properties of biological neural networks, where a group of biological neurons is interconnected to propagate an external signal to the brain. Similarly, artificial neural networks are made of units, which are often assumed simple in the sense that single numbers, their “activation” values, can describe their state.

4.1. Functionality of Artificial Neural Networks

Each unit generates an output signal based on its activation. Units are connected to each other very specifically, each connection having an individual "weight" (again described by a single number). Each unit sends its output value to all other units to which they have an outgoing connection. Through these connections, the output of one unit can influence the activations of other units. The unit receiving the connections calculates its activation by taking a weighted sum of the input signals (i.e. it multiplies each input signal with the weight that corresponds to that connection and adds these products). The output is determined by the activation function based on this activation (e.g. the unit generates output or "fires" if the activation is above a threshold value). Networks learn by changing the weights of the connections.

4.2. Over-Learning of Artificial Neural Networks

An important limitation of neural networks (and related non-linear techniques) is that of over-fitting, or over-learning. One objective when designing a neural network is to find a function, which accurately models the unknown underlying function that relates the input variables to the output variables, and to estimate this function by fitting a function to the available data points (cases). Those over-fitting graphs are often "hidden" behind a very good correlation factor, valid only for the training data. If new (test) data with unknown outcome is fed through the network, it gives wrong approximation, thus making the whole network useless for the given problem. If a function of high curvature is chosen, one can end up modeling the noise in the data, rather than the underlying function.

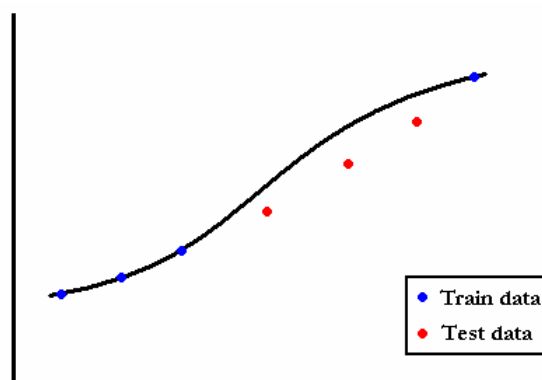


Fig. 6 – an over-fitting graph

Figure 6 illustrates this problem when fitting a curve in two dimensions - a smoother line would be better here, even if it did not pass through the data points, as it will predict results more accurately when new data is tested.

4.3. Performance of Artificial Neural Networks

The ability to perform well on new data is called *generalization*, and is the most desirable property of a neural network. The most effective approach to make a network, that generalizes well, is to hold back some of the data, and not to use it for training the network. This data can be used to cross-verify the network's performance. Cross-verification data is used in two ways. First, as network training progresses the curvature of the function increases, and so one can stop training if performance on the cross *verification* set starts to deteriorate. This prevents over-learning. Second, if one designs a number of neural networks and wishes to select the best one, they cannot safely compare the performance on the training sets, as one network may have over-learned and have an unexpectedly good performance. However, the network with the best cross-verified performance can be selected.

When the cross-verification error is used in this way, another problem may be introduced. If a large number of networks are tested, and the one with the best cross verification error selected, one is effectively performing a sampling experiment, and may end up with a network with an unusually good cross-verification performance, which does not reflect its generalization ability, but only the random sampling process. One may therefore reserve a further subset of data, the *test* set, which is used purely at the end of the design process to check that the cross verification error is not artificial. If the cross verification and test errors are close together, one may be reasonably confident that the network will generalize successfully.

5. Artificial Neural Networks in Metal Forming

Empirically measured data for certain material, within the range RT - 1200° C, are used to train a neural network, so that it can best represent the behaviour of the flow stress and to obtain smooth flow curves. Once the network is trained up to a good correlation level, it can be used in order to compute unknown values of the flow stress up to now, based on the strain, strain rate and temperature for the specified steel grade as input parameters. This is done by feeding these

parameters into the three input neurons of the network and reading the value for k_f at the output node. When performing this operation several times in order to read consecutive values of the flow stress, one can obtain data for a specified range and material and can represent them as a flow curve as shown in figure 7.

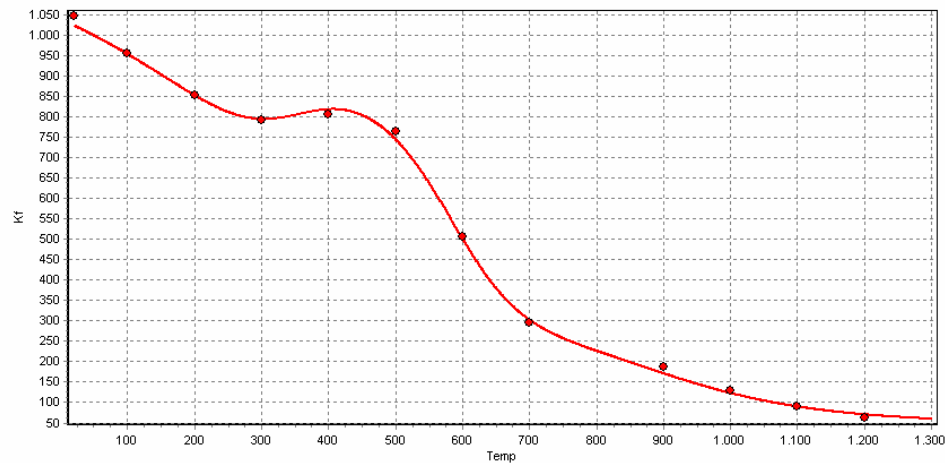


Fig. 7 – Flow curve for material 100Cr6 with $\Phi=0.5$ and $\Phi_{hip}=1.5 s^{-1}$, neural networks

As it can be seen from the graph, the neural network, trained with some empirically measured data (the red dots), has computed values that almost perfectly represent the physical behaviour of the material for cold as well as for hot forming. This graph also proves that the artificial neural networks are extremely good for interpolation, provided that the input data is dense enough. Another advantage of using neural networks is that they do not require explicit function types as the regression analysis and thus can be used for solving almost all kinds of statistical problems involving the search of a relationship between variables.

Figures 8 and 9 compare the well-known curves of Fritzsich-Siegel (left) for steel grade 100Cr6 with the ones, computed with the help of neural networks (right).

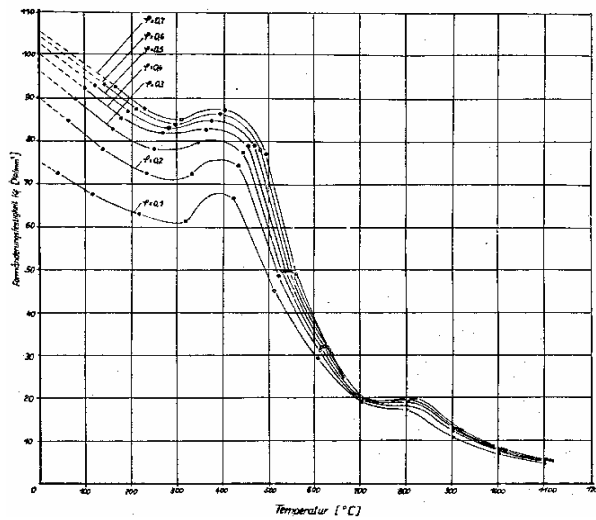


Fig. 8 – Flow curves for 100Cr6, Fritzsich-Siegel

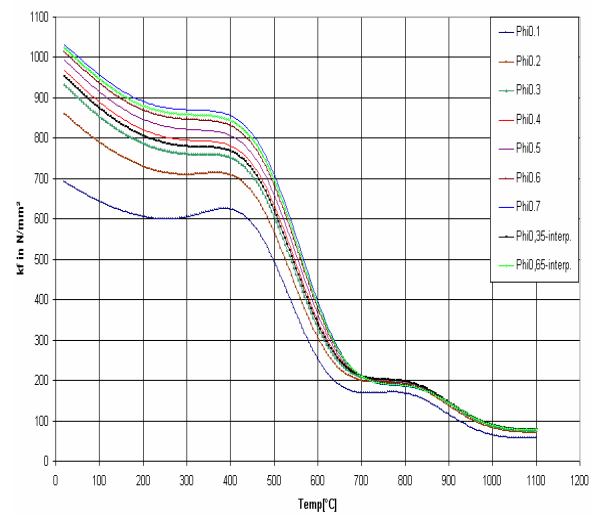


Fig. 9 – Flow curves for 100Cr6, Neural Networks

One limitation of neural networks, however, is that they require large and dense database and cannot perform well outside of this database, i.e. they are not suitable for extrapolation. Figure 10 demonstrates that the model shows unrealistic values for k_f when trying to extrapolate the input data (the red dots).

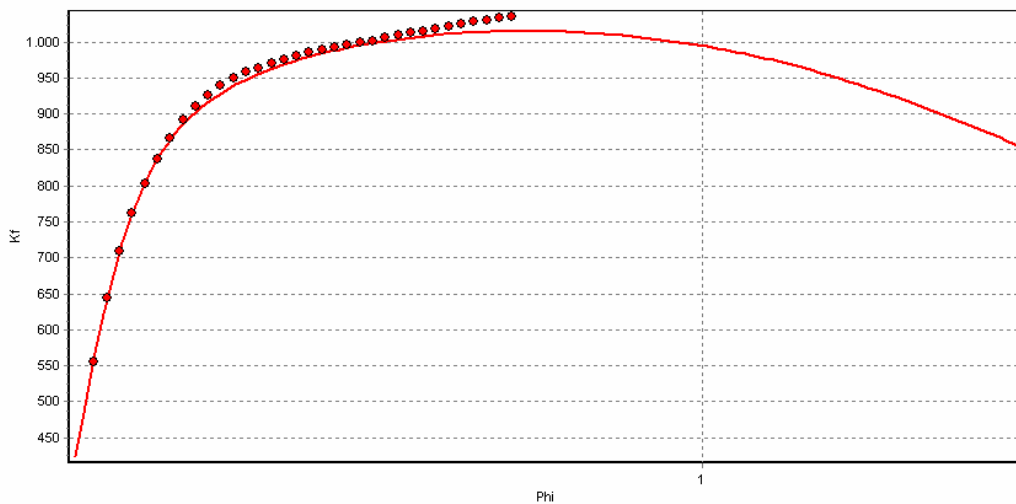
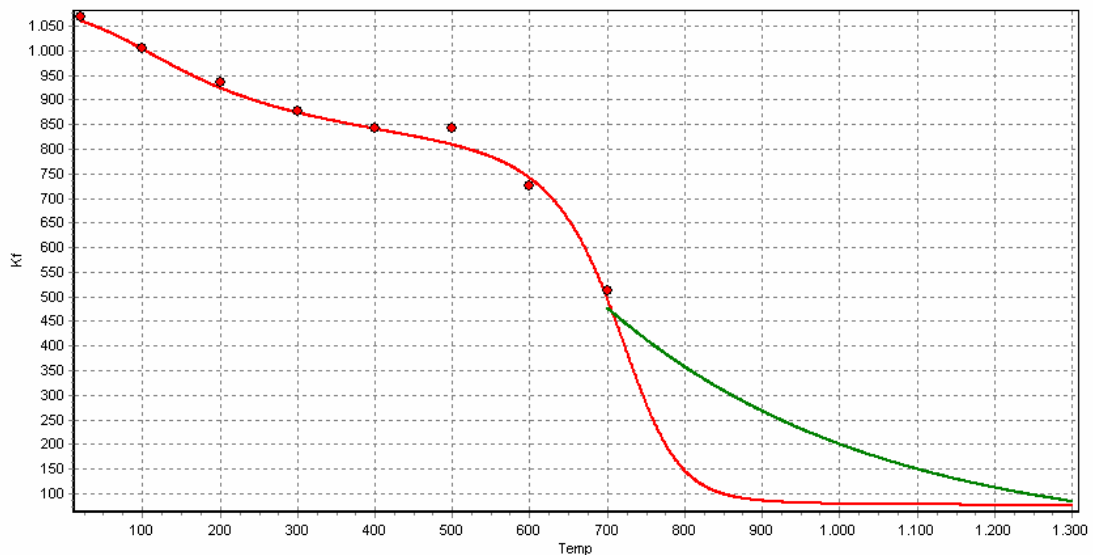


Fig. 10 – Flow curve for C60 with $\Phi_{ip}=1.5 \text{ s}^{-1}$ and $T=20^\circ \text{ C}$, extrapolation

6. Combination of Methods

Having in mind the advantages and disadvantages of both described methods – regression analysis and neural networks, one can come up with a model, that combines only the benefits of these two techniques. In figure 11 the red line demonstrates the abilities of the neural network to interpolate the values within the database (the red dots), for which the network has been trained, and to describe the physical behaviour of the material inside that region. The regression curve (the green line), on the other hand, shows more realistically the flow stress outside the database when using equation (13) on page 10 as a model function with coefficients, computed on the basis of the analysed alloy concentrations for the examined material.



*Fig. 11 – Flow curve for material C60 with $\Phi=0.5$ and $\Phi_{ip}=90 \text{ s}^{-1}$,
neural networks (red) and regression analysis (green)*

7. Realization of the Combined Model

In order to put the combined model into practice, one has to (1) train a neural network for a specified steel grade, (2) convert the trained network in the form of a recall function and then (3) call it from within software or a spreadsheet. If the database does not cover the whole desired range of values (as in figure 11), one needs to compute the coefficients of the model function, that is to be used in the regression analysis part of the combined model, and again use this function within the software or the spreadsheet.

7.1. Training a Neural Network

The first part of the routine is to gather and organize the training data and to divide it into three data sets – for training, for validation and for testing, as described in the part “Performance of Artificial Neural Networks” of this document. This division can be done manually or by using specially designed software for this purpose. The manual organization can be carried out by moving random lines of the data into any of the three sets, while keeping a certain ratio for the cases in the train, verify and test sets respectively. Furthermore, the data needs to be scaled properly in order to be used in the training of the neural networks efficiently. This routine can be very time consuming and error prone for extremely large training data and thus it is advisable to exploit the capabilities of the automation software.

7.1.1. SNNS Converter

The “SNNS Converter” is software, developed by Johannes Gottschling and the author, that converts the delimiter separated input data (text file) into the three sets, as described above (*.pat files). Figure 12 shows the main interface of the program.

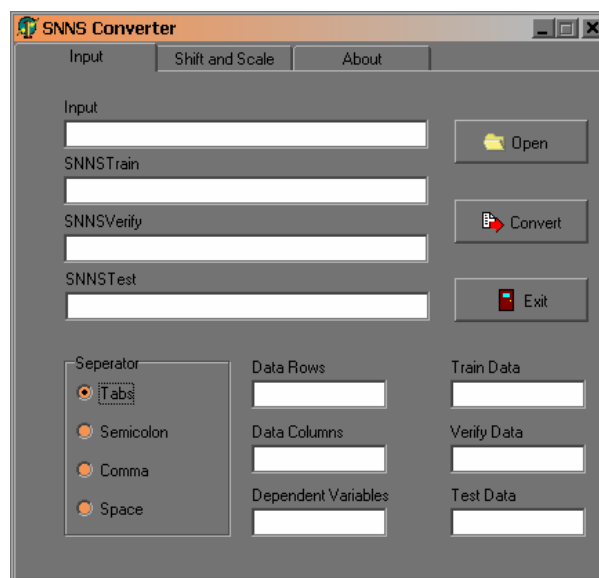


Fig. 12 – SNNS Converter, main interface

The software randomizes the order of the training cases for better performance and allows manipulation of the delimiter, the number and ratio of the sets as well as other customization

options. The software also scales and shifts the data for better efficiency when training the neural networks, using the linear function

$$y = a \cdot x - b \quad (19)$$

where the coefficients a and b are the scale and shift respectively, x is the original data and y is the converted data. The coefficients are computed by the formulae

$$a = \frac{1}{x_{\max} - x_{\min}} \quad (20)$$

$$b = \frac{-x_{\min}}{x_{\max} - x_{\min}} \quad (21)$$

The “SNNS Converter” also adds some specific header data into the three *.pat files in order to make them compatible with the software for training the neural networks, so that they can be used as input data for the latter.

7.1.2. Stuttgart Neural Network Simulator

SNNS (Stuttgart Neural Network Simulator) is a software simulator for neural networks that has the goal to create an efficient and flexible simulation environment for research on and application of neural networks [5]. Figure 13 shows parts of the interface of the program.

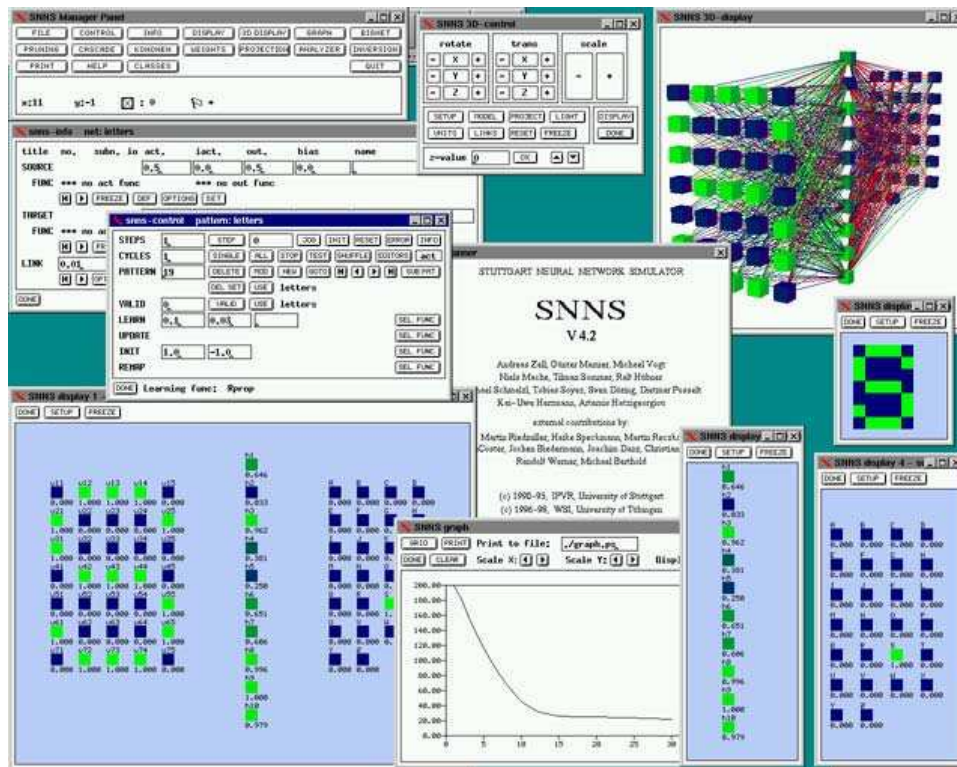


Fig.13 – Stuttgart Neural Network Simulator, parts on interface

Using the described software, one can build networks from scratch and train them with the *.pat files, created with the help of the “SNNS Converter”. Since the Stuttgart Neural Network Simulator offers several possibilities for the type of the desired network, one has to choose carefully the type that will be most suitable for the problem. It is well known from the theory of neural networks that a network may have some problems with the extrapolation of a given data collection, particularly networks from the RBF-type (Radial Basis Function Type) [4]. For this reason, networks from the MLP-type (Multi-Layer Perceptron Type) are more suitable for the described method, since they show a slightly better performance on extrapolation of the data.

Empirical evidence from extensive trial and error have shown that a nearly optimal solution for the structure of a network for the current problem is one having three input nodes, twelve nodes in the hidden layer and one output node. The three input perceptrons are used to feed the data for Phi, Phip and Temp to the network, so that a value for Kf can be observed at the output perceptron. The twelve nodes in the hidden layer serve for training the data more efficiently. A known trade-off for the number of these nodes is that the more perceptrons there are, the more flexible a network becomes, which leads to a higher possibility that it will reach a good correlation level. The high number of perceptrons, on the other hand, makes it more difficult to reach the desired performance and thus expanding the training time.

One experiment, designed to compare several possibilities for the number of hidden nodes, has shown that a network with four perceptrons has a very unstable learning curve and never reaches high correlation levels, i.e. a mean squared error of below 0.2. Having eight perceptrons slightly improves the quality of the network and also expands learning time, but still does not make the network reach sufficient values for the correlation level (mean squared error ≈ 0.2). The experiment showed that twelve nodes in the hidden layer keeps a nearly optimal trade-off for performance/learning time (mean squared error < 0.2) and thus makes it more suitable for the given problem. Sixteen nodes expand the learning time even more and do not improve the performance sufficiently enough, thus leaving the number of twelve nodes optimal for the problem at hand.

7.2. Creating a Recall Function from the Trained Network

Once the network is trained up to a good correlation level (mean squared error < 0.2) it can be saved to a *.net file by the SNNS. In order to make it possible for the network to be integrated into any software or spreadsheet, one has to transform it into a recall function by making it in the form of a DLL (Dynamic Link-Library). This recall function resembles a program that takes three input values – strain, strain rate and temperature and returns one value – that of the flow stress. A C-code generator has been created by Bernward Kett – “snns2c” (SNNS to C), that transforms the saved network (*.net file) into a usable C-function in the form of two source code files (*.c and *.h files), which are parts of the DLL, i.e. the DLL refers to them for the actual computation of the result.

Due to the nature of SNNS and snns2c, the expected input parameters of the C-function should be already scaled down using the formula $y = a \cdot x - b$, as described in the section “SNNS Converter” of this document. As a result, the output of the function is not the real value of the flow stress, but again a scaled one.

In order to make the DLL receive and return the actual values to the user, rather than the scaled ones, one has to scale the input values before feeding them into the C-function and then scale back the output value from the C-function before returning it to the user. The scaling back is done by using the inverse function of $y = a \cdot x - b$, namely

$$x_o = \frac{y_o + b}{a} \quad (22)$$

where y_o is the scaled output of the C-function and x_o is the actual value of the flow stress, which is to be returned to the user.

Figure 14 uses the standard UML notation to show a sequence diagram for the flow of the variables within the DLL and the computation of the flow stress, based on the input from the user (for explanation of the diagram elements, please refer to Appendix I). Alternatively, figure 15 illustrates the same flow within the recall function, using a notation, developed by the author. The latter diagram, in contrast to the UML notation, does not show the time constraints, but illustrates the use of neural networks for the computation of the flow stress.

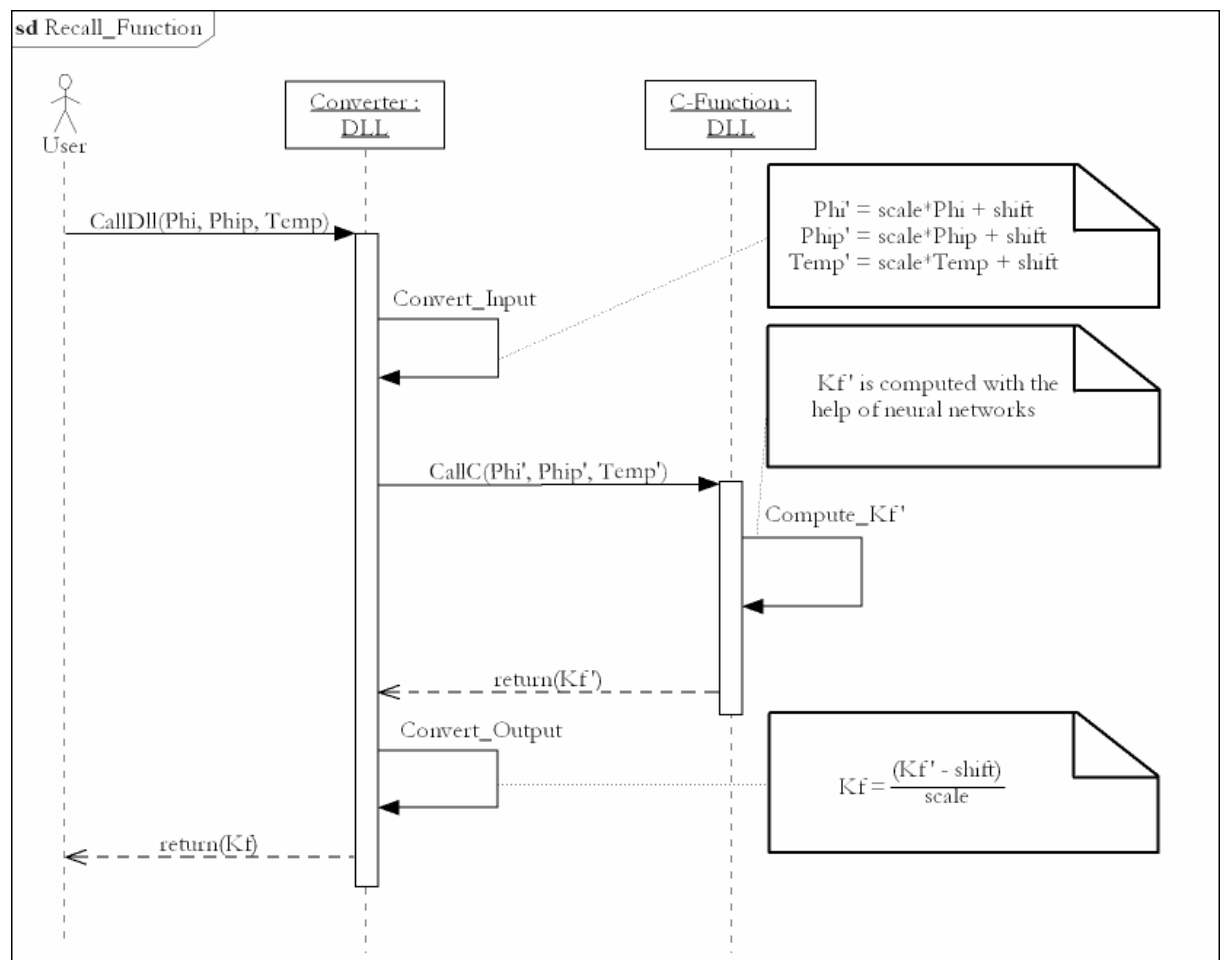


Fig.14 – UML 2.0 Sequence diagram for the recall function

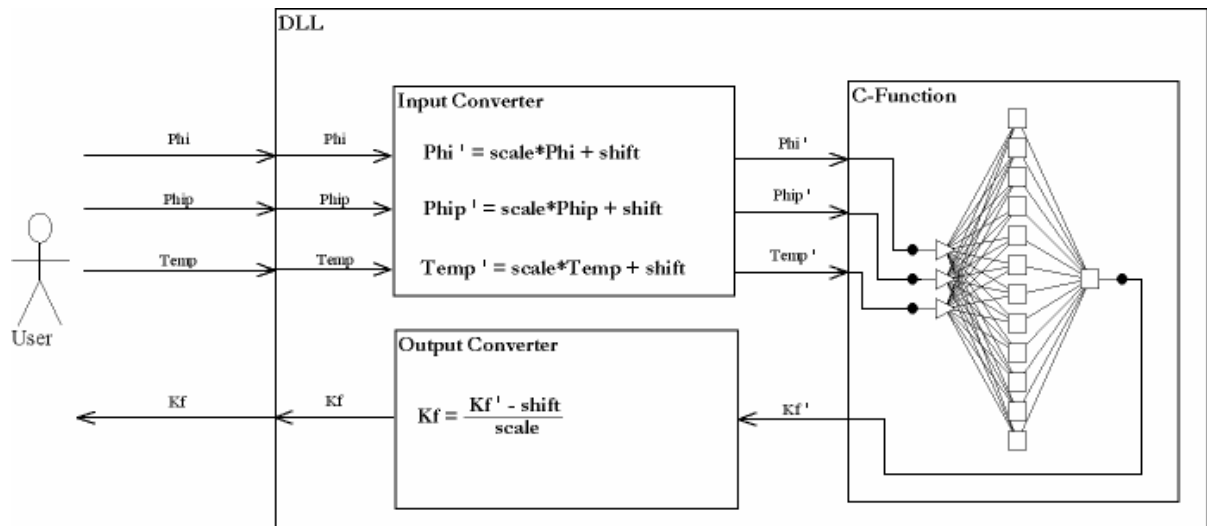


Fig. 15 – Diagram of the flow of control within the DLL

When the neural network is trained by SNNS and converted by the snns2c, the two generated source code files have to be renamed for better consistency. After renaming, the files have to be modified for the environment, in which they will be compiled and lastly put into a project and compiled to a DLL using for example Microsoft® Visual C++ 6.0 (please refer to Appendix II for detailed description about building the DLL).

7.3. Calling the DLL from within software

Once the DLL for a specified material is created, it can be used by any software or spreadsheet in order to compute values of the flow stress based on three input parameters – strain, strain rate and temperature. Moreover, continuous graphs (two-dimensional as well as three-dimensional) can be plotted via a series of subsequent calls to the DLL of the corresponding steel grade and reading the values returned from this recall function. In this way, the dependency of the flow stress on any of the three input parameters can be observed with the help of the plotted graph(s).

7.3.1. “Flow Stress Plotter”

“Flow Stress Plotter” is software, written by the author, that displays the dependency of the flow stress on the strain, strain rate or temperature for a given material in a graphical way. It allows plotting this relation in two-dimensional graph as well as in three-dimensional surface and calculating single values of the flow stress, based on the above parameters. All computations are carried out using neural networks and regression analysis.

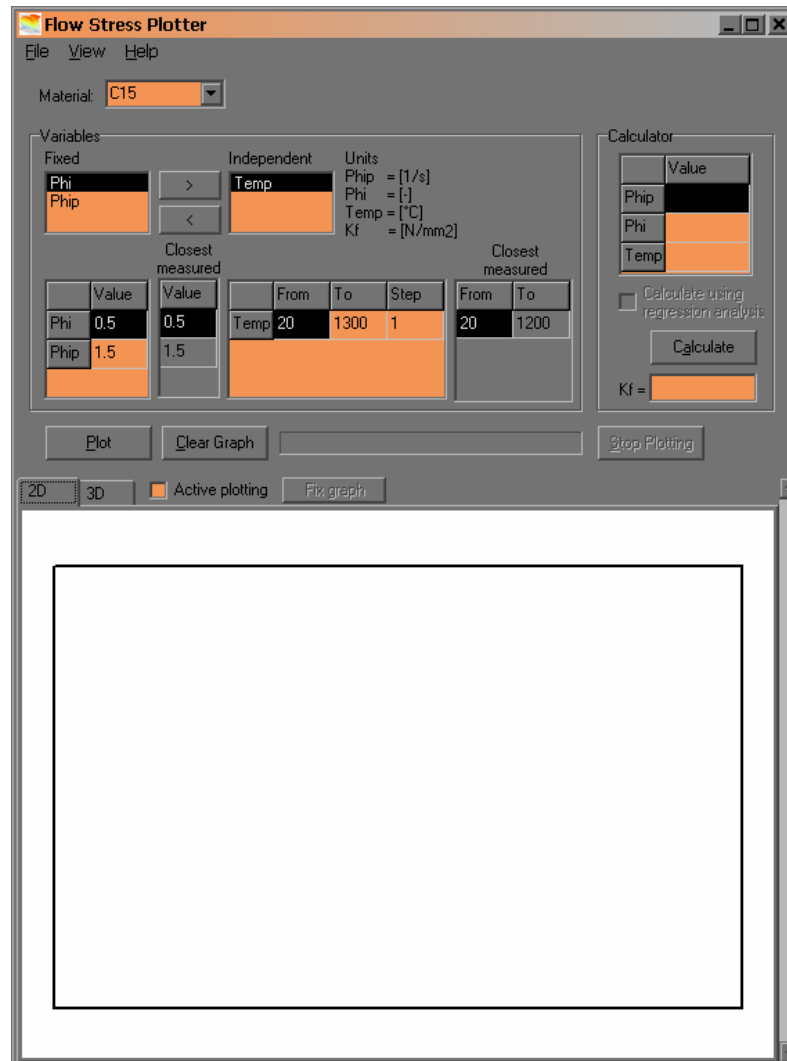


Fig. 16 – Flow Stress Plotter, main interface

The software uses the already described recall functions in the form of DLLs to display information about selected steel grades. Since the DLLs represent trained neural networks, it is possible that the displayed flow curve does not represent realistically the behaviour of the material. Such situation may occur when the specified parameters are (far) outside the database, which served for training the network. For this reason, additional tables are introduced to display

measured values in the database that are closest to the specified ones, so that the user can decide how true the flow curve is represented. On the other hand, when the specified parameters exactly match the training data (the measured points), this data is also displayed on the graph in the form of single dots, so that the user can observe how well are the neural networks trained.

Additional features include active plotting – observing the change of the flow curve upon changing the parameters in real time; exporting the points of plotted graphs in many file formats in a form, consistent with other software solutions (Microsoft® Excel®, Word®, etc.); exporting the canvas with the plotted graphs to different picture file formats; and many more.

7.3.2. Usability of “Flow Stress Plotter”

One of the main advantages of the software is that it has a user-friendly interface and functionality easy to navigate. Moreover, all the features are explained in the user manual (please refer to Appendix III for this manual) in brief as well as in detailed way accessible to all kinds of computer users, familiar with this problem space – beginners, advanced and expert users. The software also allows new recall functions to be created independently of the program version and location and to be added at run time, allowing uninterrupted use.

“Flow Stress Plotter” covers also the main usability issues, such as *learnability* – the system is easy to learn and the user can rapidly start working with it; *efficiency* – once the user has learned the software, he or she can achieve high level of productivity; *memorability* – the system is easy to remember, so that the casual user does not need to re-learn all the functionality upon each use; *error rate* – the system has a low error rate, so that users make few errors when using the system, and if they do make errors, they can easily recover from them, due to the self-explanatory error messages. Furthermore, catastrophic errors are not reported to have occurred.

Almost all dialog elements of the software have tooltips that explain in brief the functionality of these elements, so that the advanced user does not need to refer to the manual in order to accomplish his or her tasks. Moreover, the system allows that the expert computer user can work with the system without the need of any additional help, such as manuals or tooltips.

8. Conclusion

The isolated computation of cold and hot flow curves, using the well-known mathematical functions, is not suitable for wide temperature ranges and makes it difficult to obtain values of the flow stress in the boundary region between both curves. When used alone, regression analysis fails to describe the physical behaviour of a material in a flow stress diagram for temperatures less than 800° C, since no suitable model function is known up to now that covers the whole range RT – 1200°. Artificial neural networks tend to show more realistic results for the computation of the flow stress in that region, depending, however, on large and dense database.

When combining the advantages of artificial neural networks and regression analysis, one can come up with a model, that allows computation of the flow stress for certain material, based on the strain, strain rate and temperature for the complete region 20° – 1200° C. This model can be utilized in practice by compiling a DLL for the specified steel grade and using the specially designed software for this purpose – the “Flow Stress Plotter”.

References

[1] – www.metal-mart.com/Dictionary/dictlist.htm

[2] – Mauk, P.J., Gottschling, J.: “Mathematical Methods to Ascertain Flow-Curve Functions for Cold and Hot Forming”, Duisburg, 2006

[3] – en.wikipedia.org

[4] – Gottschling, J., Mauk, P.J.: “Hot Flow Curves of Metallic Materials”, Duisburg, 2006

[5] – www-ra.informatik.uni-tuebingen.de/SNNS/announce.html

[6] – Gottschling, J., Weyh, B.: “Computer Based Engineering Mathematics”, Duisburg, 2005

Nielsen, J.: “Usability Engineering”, 1993

Sykes, A.: “An Introduction to Regression Analysis”, Chicago

Fox, J.: “Nonlinear Regression and Nonlinear Least Squares”, 2002

Smyth, G.: “Nonlinear Regression”, Chichester, 2002

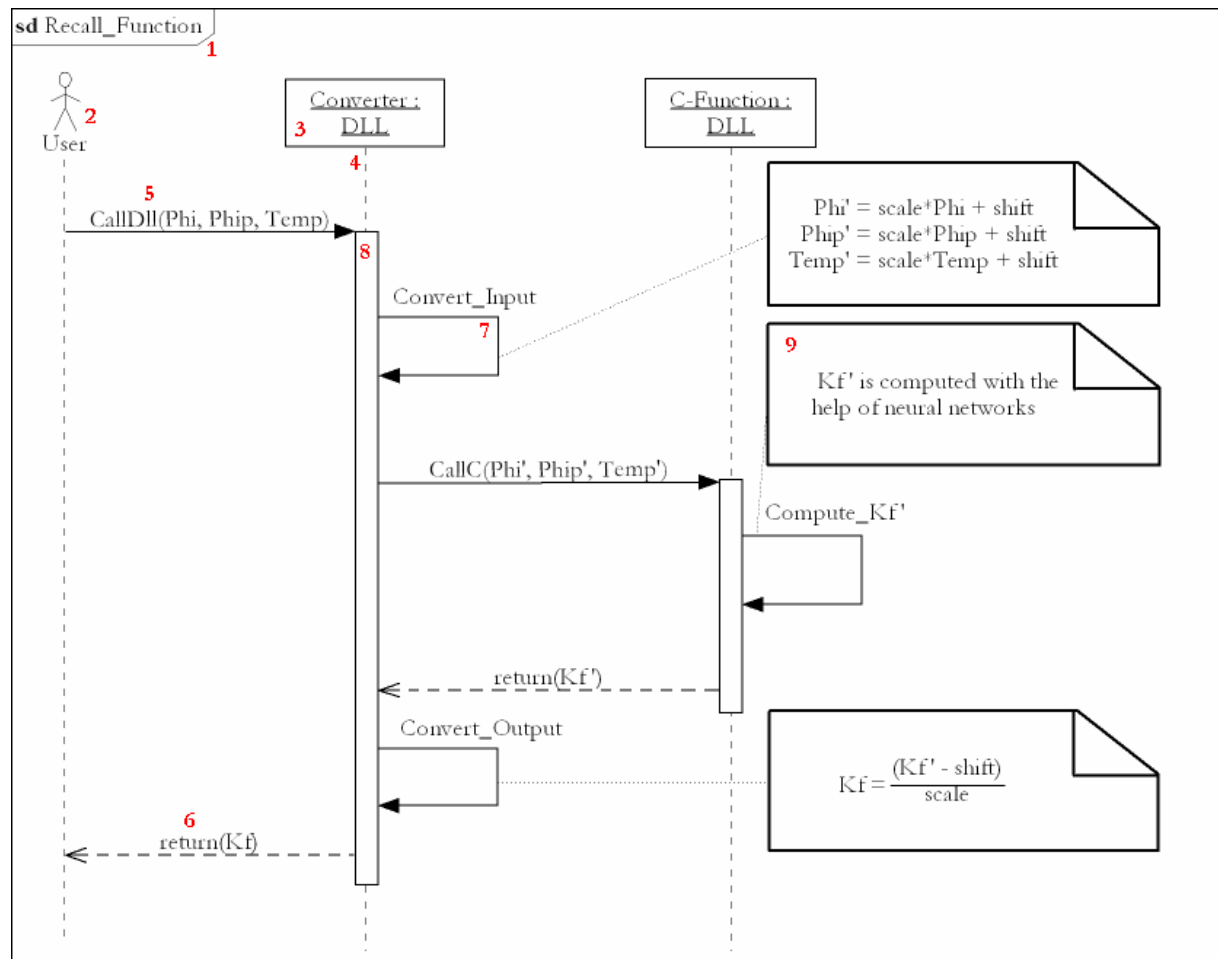
Vinsnes, A.G.: “Regression Analysis”, 2001

elsa.berkeley.edu/sst/regression.html

www.nlreg.com/intro.htm

Statistica[®], User’s Manual

Appendix I – Explanation of the elements of the UML 2.0 Sequence diagram



Legend:

1 – Name of the sequence diagram. In this case – “Recall_Function”.

2 – Actor. In this case – the user of the function.

3 – Object. In this case – the component, that converts the input and output.

4 – Lifeline of the object. Specifies the duration, for which the object is available.

5 – Synchronous message call. The sender is bound to wait for response from the called object.

In this case the user sends the message CallDll(Phi, Phip, Temp) to the Input Converter and

waits for its output – k_f .

6 – Synchronous message reply. The object is replying to the caller's synchronous message call.

In this case the converter is returning the value of k_f upon input of the three parameters.

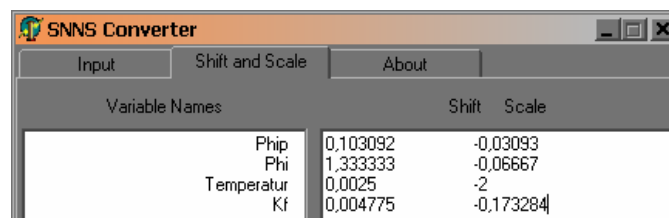
7 – Self message. The object is executing the message internally. In this case "Convert_Input" is done inside the Converter object.

8 – Execution. Shows the amount of time, for which the object is processing the caller's message.

9 – Note. Explains events informally.

Appendix II – Detailed Steps for Producing a DLL from Training Data

1. Compute the scale and shift, on the basis of the training data (e.g. “C60.txt”), for the three inputs – Phip, Phi and Temp and the output – Kf by the formulae 20 and 21 as described in the section “SNNS Converter” of this document.
2. Using the SNNS Converter, open the training data and input the computed scale and shift in the right edit box of the tab “Shift and Scale” of the software. The values should follow the sequence of the variables to the left and should be separated by one TAB only. Figure 2.1 illustrates the correct placement of the values.



Variable Names	Shift	Scale
Phip	0,103092	-0,03093
Phi	1,333333	-0,06667
Temperatur	0,0025	-2
Kf	0,004775	-0,173284

Fig. 2.1 – SNNS Converter, shift and scale

- Press the convert button in the Input tab of the software in order to generate three *.pat files, that will be used in the training of the neural network.
3. Open the Stuttgart Neural Network Simulator (SNNS) and select Tools → Create → Layers..., change the Height to 3, the Unit Type to Input and press Create; change the Height to 12, the Unit Type to Hidden and press Create; change the Height to 1, the unit type to Output and press Create and then Close. Select Tools → Create → Connections..., select the second radio button “Connect feed-forward”, unmark “With shortcut connections” if marked, press connect and then Close.
 4. Choose File → Open and select two of the three generated *.pat files, namely those, which end in “train” and “verify”. The “test” file is not needed for this purpose. Select View → Error Graph and leave it opened. Choose Tools → Control Panel → Patterns and select the file, which ends in “train” for the training set and the other for the validation set (e.g. “C60train” for training set and “C60verify” for validation set). Switch to the Learning tab and select the “Backprop-Momentum” learning function. Change the value of μ to 0.2, that of d_{\max} to 0, the Cycles to 5000, mark the Shuffle checkbox and press Init.

5. Press Learn All to start training the network with the training data. Repeat this step until the error graph shows error rate, that is as far below 0.2 as possible. This ensures a well-trained neural network. You can reduce the y-axis range for better precision when reading the error by pressing the second button in the top left part of the error graph.
6. Once the trained network has a low error rate (below 0.2), choose File → Save and specify a name, which starts with an “n”, standing for “network”, then followed by the name of the steel grade (e.g. “C60”) and ending in any letter or sequence of letters (e.g. “kw”)¹ in order to distinguish it from the project name (the name of the dll). For example, the name of the trained network for material C60 is nC60kw.
7. Run the snns2c converter on the saved network (*.net) in order to generate the two source code files (*.c and *.h) for the recall function. This is done by calling the snns2c with the *.net file as first parameter:

```
snns2c.exe <netfile>
```

For example if the network is nC60kw, the command should be:

```
snns2c.exe nC60kw.net
```

provided that the *.net file is in the same directory as the executable. Otherwise, the correct path should be specified.

8. Rename the two generated source files, so that they have the same name as the *.net file. For example if you generated the source files from network nC60kw.net, the two files should be named nC60kw.c and nC60kw.h.

Note: Steps 9 and 10 can also be performed after step 14 using the Microsoft[®] Visual C++ 6.0 source code editor.

1. “kw” initially stand for “kalt/warm” (in English: “cold/warm”) to indicate, that the network was trained for both cold and warm temperature regions. Now no meaning is attributed to the letters and they stand only to resolve the ambiguity with file names.

9. Open the *.c file in any text or source code editor and replace the first source line in the *.c file from “#include <math.h>” to “#include stdafx.h”. Modify the name of the main function (near the end of the file) to the same name as the file (e.g. “nC60kw”). For example the main function for material C60 looks like this:

```
int nC60kw(float *in, float *out, int init)
```

10. Open the *.h file in any text or source code editor and change the “extern” function to the same name as the file (as in the previous step). For example the extern function for material C60 looks like this:

```
extern int nC60kw(float *in, float *out, int init);
```

Change the name of the static structure (the beginning of the last line) to the above described name and add “REC” in the end, standing for “record” (e.g. “nC60kwREC”).

Change the initialization of the third element from the structure (the end of the last line) to the one in step 9. For example the last line of the *.h file for material C60 looks like this:

```
} nC60kwREC = {3, 1, nC60kw};
```

11. Open Microsoft® Visual C++ 6.0 and select File → New → Projects → Win32 Dynamic-Link Library. Specify a project name, starting with an “n” and followed by the name of the material (example for name of the project is “nC60”) and click “OK”. Select the second radio button - “A simple DLL project”, click “Finish” and confirm the new project information window with “OK”.
12. Select the two newly created *.cpp files (e.g. “nC60.cpp” and “StdAfx.cpp”) in the “Source Files” folder of the workspace (the left panel) and press DEL on the keyboard in order to remove them from the project.
13. Externally, change the extensions of the above two files from *.cpp to *.c in order to make them compatible with the “C” source code notation.
14. Again in the editor, select Project → Add To Project → Files... and choose the four modified files – the two files from step 13 (e.g. “nC60.c” and “StdAfx.c”) and the two files, containing the source code for the neural network (e.g. “nC60kw.c” and “nC60kw.h”).

15. Select Build → Set Active Configuration... and set the active configuration to “Win32 Release”. Select Project → Settings... → C/C++, choose the category “Precompiled Headers” and select the first radio button “Not using precompiled headers”. Switch to the “Link” tab and in the “project options” edit box add the following statement at the end of the box: “/EXPORT:nnrecall” (without the quotes) in order to make it possible for other programs to call this dll. Confirm your changes with “OK”.
16. Open the file, containing the DllMain function (e.g. “nC60.c”) and add the following code (copy and paste it) at the end of the file (outside the DllMain function):

```
double __stdcall nnrecall(char *kfname, double phip, double phi, double
temp, double kf)
{
    float *netInput, *netOutput;
    netInput = malloc(3*sizeof(float));
    netOutput = malloc(1*sizeof(float));

    // Input additional code here...

    free(netInput);
    free(netOutput);
    return(kf);
}
```

17. Replace the comment line “// Input additional code here...” with source code, based on the following *sample* code:

```
if (strcmp(kfname, "material_name") == 0) {
    netInput[0] = phip_scale*phip - phip_shift;
    netInput[1] = phi_scale *phi - phi_shift;
    netInput[2] = temp_scale *temp - temp_shift;
    material_source_name(netInput, netOutput, 0);
    kf = (netOutput[0] + kf_shift) / kf_scale; }
}
```

where the words in *italics* mean the following:

material_name is the name of the steel grade (e.g. “C60”)

phip_scale is the scale, computed with formula 20 for Phip

phip_shift is the shift, computed with formula 21 for Phip

phi_scale is the scale, computed with formula 20 for Phi

phi_shift is the shift, computed with formula 21 for Phi

temp_scale is the scale, computed with formula 20 for Temp

temp_shift is the shift, computed with formula 21 for Temp

kf_scale is the scale, computed with formula 20 for Kf

kf_shift is the shift, computed with formula 21 for Kf

material_source_name is the name of the source file of the neural network (e.g. “nC60kw”)

For example the source code for material C60 looks like this:

```
if (strcmp(kfname, "C60") == 0) {  
    netInput[0] = 0.01112*phip - 0.001112;  
    netInput[1] = 1.315789*phi - 0.05263;  
    netInput[2] = 0.0009259*temp - 0.01852;  
    nC60kw(netInput, netOutput, 0);  
    kf = (netOutput[0] + 0.07146) / 0.0009919; }  
}
```

18. Press F7 to compile and build the DLL.

Note: Ignore the compiler warnings.

If you have followed all steps and met no errors while compiling and building the DLL, you can find this newly created DLL in the “Release” folder of your project.

University Duisburg-Essen

Flow Stress Plotter v1.5

Appendix III – User's Manual

Contents

	Introduction.....	2
I.	Plotting a 2D graph.....	2
II.	Plotting a 3D graph.....	4
III.	Using the calculator.....	6
IV.	Exporting a 2D or 3D graph as a bitmap.....	7
V.	Exporting the points of a 2D graph to a file.....	8
VI.	Exporting the points of a 3D graph to a file.....	9
VII.	Using the 2D view.....	10
VIII.	Using the 3D view.....	11
IX.	Adding a new material to the database.....	12
X.	Active Plotting.....	12
XI.	Closest Measured points.....	13
XII.	Stop Plotting.....	13

Introduction

Flow Stress Plotter is software that displays the dependency of the flow stress on the strain, strain rate or temperature for a given material in a graphical way. It allows plotting this relation in two-dimensional graphs as well as in three-dimensional surfaces and calculating single values of the flow stress, based on the above parameters. All computations are carried out using neural networks and regression analysis.

I. Plotting a 2D graph

Brief steps for plotting a two dimensional graph:

1. Choose a material.
2. Choose one *Independent* variable.
3. Fill in the range of values for the *Independent* variable (From, To, Step).
4. Fill in the values of the *Fixed* variables.
5. Press the button *Plot*.

Detailed steps for plotting a two dimensional graph:

1. Choose a material:

Choose the name of the desired material from the combo box next to the label *Material* either by typing the material name using the keyboard, or by selecting it from the drop-down list.

2. Choose one *Independent* variable:

Use the buttons “<” and “>” between the two list boxes in order to rearrange the variables into the desired places. The “>” button will move the selected *Fixed* variable to the list of the *Independent* variables. The “<” button will move the selected *Independent* variable to the list of the *Fixed* variables. Additionally, drag and drop or double click can be used for easier navigation instead of the buttons “<” and “>”.

3. Fill in the range of values for the *Independent* variable (*From*, *To*, *Step*):

Use the keyboard to enter values in the table under the *Independent* list box. The value under *From* is the initial (or starting) value for the horizontal axis of the graph. The value under *To* is the final (or ending) value for the horizontal axis of the graph. The value under *Step* specifies the minimum distance between two consecutive values on the horizontal axis of the graph.

Typing the *From* value and pressing enter will automatically move to the *To* value. Typing the *To* value and pressing enter will automatically move to the *Step* value and will calculate the default step using the formula $(To - From)/100$.

Note: *From* value should be in the range $0 < From < To$; *To* value should be in the range $From < To$; *Step* value should be in the range $0 < Step < To - From$.

4. Fill in the values of the *Fixed* variables:

Use the keyboard to enter values in the table under the *Fixed* list box. The cells under *Value* specify the values of the *Fixed* variables. Typing a value in the upper cell and pressing enter will automatically move to the lower cell and vice versa.

5. Press the button *Plot*:

Press the button *Plot*, or use the shortcut ALT+P in order to plot a two dimensional graph k_f vs. *Independent* variable, where the vertical axis shows the k_f values and the horizontal axis shows the *Independent* values.

If the *Independent* variable is *Temp*, an additional graph is plotted (for values $> 800^\circ\text{C}$), that is computed using the regression analysis.

If empirically measured data for the specified parameters are present, they are also plotted on the canvas.

Note: While calculating (plotting) the graph, all dialog elements are disabled in order to prevent interference in computation and to carry it out in the shortest amount of time. You can stop this action by pressing the button *Stop Plotting*

II. Plotting a 3D graph

Brief steps for plotting a three dimensional graph:

1. Choose a material.
2. Choose two *Independent* variables.
3. Fill in the range of values for the *Independent* variables (From, To, Step).
4. Fill in the value of the *Fixed* variable.
5. Press the button *Plot*.

Detailed steps for plotting a three dimensional graph:

1. Choose a material:

Choose the name of the desired material from the combo box next to the label *Material* either by typing the material name using the keyboard, or by selecting it from the drop-down list.

2. Choose two *Independent* variables:

Use the buttons “<” and “>” between the two list boxes in order to rearrange the variables into the desired places. The “>” button will move the selected *Fixed* variable to the list of the *Independent* variables. The “<” button will move the selected *Independent* variable to the list of the *Fixed* variables. Additionally, drag and drop or double click can be used for easier navigation instead of the buttons “<” and “>”.

3. Fill in the range of values for the *Independent* variables (*From*, *To*, *Step*):

Use the keyboard to enter values in the table under the *Independent* list box. The values under *From* are the initial (or starting) values for the corresponding horizontal axis of the graph. The values under *To* are the final (or ending) values for the corresponding horizontal axis of the graph. The values under *Step* specify the minimum distance between two consecutive values on the corresponding horizontal axis of the graph.

Typing the *From* value and pressing enter will automatically move to the *To* value for the corresponding *Independent* variable. Typing the *To* value and pressing enter will automatically move to the *Step* value for the corresponding *Independent* variable and will calculate the default step using the formula $(To - From)/100$.

Note: *From* values should be in the range $0 < From < To$; *To* values should be in the range $From < To$; *Step* values should be in the range $0 < Step < To - From$.

4. Fill in the value of the *Fixed* variable:

Use the keyboard to enter a value in the table under the *Fixed* list box. The cell under *Value* specifies the value of the *Fixed* variable.

5. Press the button *Plot*:

Press the button *Plot*, or use the shortcut ALT+P in order to plot a three dimensional graph k_f vs. first *Independent* variable vs. second *Independent* variable, where the vertical axis shows the k_f values and the two horizontal axes show the values of the two *Independent* variables.

Note: While calculating (plotting) the graph, all dialog elements are disabled in order to prevent interference in computation and to carry it out in the shortest amount of time. You can stop this action by pressing the button *Stop Plotting*

III. Using the calculator

Brief steps for using the calculator:

1. Choose a material.
2. Fill in the values for $Phip$, Phi and $Temp$.
3. Specify whether or not to use regression analysis.
4. Press the button *Calculate*.

Detailed steps for plotting a three dimensional graph:

1. Choose a material:

Choose the name of the desired material from the combo box next to the label *Material* either by typing the material name using the keyboard, or by selecting it from the drop-down list.

2. Fill in the values for $Phip$, Phi and $Temp$.

Use the keyboard to enter the values for $Phip$, Phi and $Temp$. Typing a value and pressing enter will automatically move to the next cell.

3. Specify whether or not to use regression analysis.

If the value for $Temp$ is less than 800, the checkbox *Calculate using regression analysis* is disabled (by default). If the value is greater or equal to 800, the checkbox is enabled and checked.

If the box is checked, the k_f value will be calculated using the formula:

$$K_f = M_0 \cdot e^{M_1 T} \cdot \dot{\varphi}^{M_2 + M_5 T} \cdot \varphi^{M_3} \cdot e^{M_4 \varphi}$$

where M_0 through M_5 are regression coefficients, φ is Phi , $\dot{\varphi}$ is $Phip$ and T is the $Temp$.

4. Press the button *Calculate*:

Press the button *Calculate*, or use the shortcut ALT+A in order to calculate the value of k_f .

IV. Exporting a 2D or 3D graph as a bitmap

Brief steps for exporting a two or three dimensional graph as a bitmap:

Assumption: the desired graph is already plotted and is in view.

1. Select *File->Save Graph*.
2. Write the name and path of the desired bitmap and choose extension.
3. Press the button *Save*.

Detailed steps for exporting a two or three dimensional graph as a bitmap:

Assumption: the desired graph is already plotted and is in view.

1. Select *File->Save Graph*.

Select the menu item *Save Graph* under the menu *File*, or press the shortcut CTRL+S.

2. Write the name and path of the desired bitmap and choose extension.

By default, the software suggests that the file name is “FlowStress_<date>”, where “<date>” stands for today’s date; the default path is the directory where the executable of the software is located; and the default extension is *bmp*. You can change any of the default parameters by typing the desired ones using the keyboard.

3. Press the button *Save*.

Press the button *Save*, or use the shortcut ALT+S in order to save the graph as a bitmap in the specified location.

V. Exporting the points of a 2D graph to a file

Brief steps for exporting the plotted points of a two dimensional graph to a file:

Assumption: the desired graph is already plotted and is in view.

1. Select *Export Points* from the context menu of the desired graph.
2. Write the name and path of the desired file and choose extension.
3. Choose a delimiter.
4. Press the button *Save*.

Detailed steps for exporting the plotted points of a two dimensional graph to a file:

Assumption: the desired graph is already plotted and is in view.

1. Select *Export Points* from the context menu of the desired graph.

Press the right mouse button over the graph that you want to export the points of and choose *Export Points* or press the key button “e”.

Note: If you do not see the option *Export Points*, that means that you have not positioned the cursor exactly over the graph. Try zooming in for a better precision. Furthermore, you cannot export *measured* points.

2. Write the name and path of the desired file and choose extension.

You can either write the name and path using the keyboard, or press the button in the right part of the edit box (left of the button *Save*) to open the *Browse* dialog, where you can select the path and extension using the mouse (then confirm the changes using the button *Open*).

3. Choose a delimiter.

After selecting the file name and extension, the software automatically chooses the proper delimiter. You can change it by using the mouse.

Note: By default, the *xls* format requires that the delimiter is *TAB* and *csv* format requires that the delimiter is *Semicolon* in order to open the file using Microsoft® Excel® properly.

4. Press the button *Save*.

Press the button *Save*, or use the shortcut ALT+S in order to export the plotted points of the graph to a file in the specified location.

VI. Exporting the points of a 3D graph to a file

Brief steps for exporting the plotted points of a three dimensional graph to a file:

Assumption: the desired graph is already plotted and is in view.

1. Press *Export Points* button above the plotting area.
2. Write the name and path of the desired file and choose extension.
3. Choose a delimiter.
4. Press the button *Save*.

Detailed steps for exporting the plotted points of a two dimensional graph to a file:

Assumption: the desired graph is already plotted and is in view.

1. Press *Export Points* button above the plotting area.

Press the button *Export Points* in order to bring up a dialog that allows you to modify the format, in which the data will be saved.

Note: The *Export Points* feature of three dimensional graphs saves the points only of the last plotted graph. Exporting the points of previously drawn graphs is not yet possible.

2. Write the name and path of the desired file and choose extension.

You can either write the name and path using the keyboard, or press the button in the right part of the edit box (left of the button *Save*) to open the *Browse* dialog, where you can select the path and extension using the mouse (then confirm the changes using the button *Open*).

3. Choose a delimiter.

After selecting the file name and extension, the software automatically chooses the proper delimiter. You can change it by using the mouse.

Note: By default, the *xls* format requires that the delimiter is *TAB* and *csv* format requires that the delimiter is *Semicolon* in order to open the file using Microsoft® Excel® properly.

4. Press the button *Save*.

Press the button *Save*, or use the shortcut ALT+S in order to export the plotted points of the graph to a file in the specified location.

VI. Using the 2D view

Possible actions in the two dimensional plotting area (canvas):

- Hide all plotted graphs (Right mouse button on an empty area -> *Hide All*)
- Show all plotted graphs (Right mouse button on an empty area -> *Show All*)
- Delete all plotted graphs (Right mouse button on an empty area -> *Delete All*)
- Restore the view (Right mouse button on an empty area -> *Restore View* or double click with the left mouse button)
- Zoom in/out (Mouse Scroller Up/Down or the Right Scroll Bar Up/Down)
- Move the view (Click and hold the right mouse button and move the mouse)

Possible actions on any plotted two dimensional graph (except on measured points):

- Hide the graph (Right mouse button on the graph -> *Hide this*)
- Show the graph alone (Right mouse button on the graph -> *Show only this*)
- Export the plotted points (Right mouse button on the graph -> *Export points*)
- Delete the graph (Right mouse button on the graph -> *Delete this*)
- View the values of particular points (Position the cursor over the desired point)

Note: The legend in the lowermost part of the window shows all the plotted graphs, where each graph can be switched on or off by using the check boxes to the left of the labels.

“(regression)” specifies that the corresponding graph is computed using the regression analysis and the formula:

$$K_f = M_0 \cdot e^{M_1 T} \cdot \dot{\varphi}^{M_2 + M_5 T} \cdot \varphi^{M_3} \cdot e^{M_4 \varphi}$$

“(measured)” specifies that the points represent empirically measured data for the chosen parameters.

VII. Using the 3D view

Possible actions in the three dimensional view:

- Tilt the plotted graph(s) (Click and hold the left mouse button and move the mouse up or down)
- Pan the plotted graph(s) (Click and hold the left mouse button and move the mouse left or right)
- Move the view (Click and hold the right mouse button and move the mouse)
- Enable/Disable *Live Rotation* (Check/Uncheck the *Live Rotation* box)

Note: When enabled, *Live Rotation* allows the three dimensional graph to be smoothly rotated in real time, which can become very slow for graphs consisting of many points (usually more than 200.000). If the plotted graph consists of more than 200.000 points, *Live Rotation* is automatically switched off in order to prevent low frame rates. When *Live Rotation* is disabled, the user can still rotate the graph, however not in real time, i.e. only the walls of the graph are rotated and after the rotation ends (the user releases the left mouse button), the complete graph is repainted.

VIII. Adding a new material to the database

Adding a new material to the database is done by selecting the menu option *Import to DataBase* under the *File* menu. A dialog opens, that allows you to select the path and file name of the files, which you want to add to the database.

Note: Please refer to Appendix IV for the file conventions

IX. Active Plotting

Active plotting allows you to observe the change of the flow curve upon changing the parameters in real time. In order to use this feature, select the checkbox *Active Plotting*, press the left mouse button on any *Fixed* or *Independent* value and start dragging the mouse up or down or rolling the mouse scroll button up or down in order to increase or decrease respectively the selected value. You can also edit the values as usual using the keyboard and upon pressing *enter*, the graph will be re-drawn using the currently specified parameters. When you are satisfied with the parameters of the graph, you can press *Fix Graph* or deselect *Active Plotting* in order to make the changes permanent and finish the adjustment of the values.

Note: This feature is available only for two dimensional graphs. You may want to use *Active Plotting* mainly with *Fixed* variables, since the active changing of the *Independent* variables will barely make any visible changes.

XI. Closest Measured points

The two tables, containing the closest measured points show the measured points that are closest to the specified ones in the respective tables to the left. Upon changing the values of the *Fixed* or *Independent* parameters, the closest measured points are re-calculated and shown in the respective tables. You can use them to decide how true the flow curve is represented, i.e. if the difference between the specified values and the closest measured ones becomes greater this proposes a non-realistic plotting of the flow curve. On the other hand, when the specified parameters exactly match the training data (the measured points), this data is also displayed on the graph in the form of single dots, so that the user can observe how well are the neural networks trained.

Note: You can view all the measured points for a steel grade if you select that material from the list and then choose *Measured Points* under the *View* menu item, or press the shortcut CTRL+M.

XII. Stop Plotting

Pressing the button *Stop Plotting* causes the computing of the graph to cease and cancels its drawing.

Note: This button is enabled only during actual computation.

Appendix IV – File Conventions for Compatibility with the Flow Stress Plotter

Names of Recall Functions (*.dll)

1. Should start with the letter “n”
2. Should be followed by the steel grade’s name
3. Any non-alphanumeric characters should be replaced with underscores (“_”)

e.g.: “nX10Cr13.dll”, “n28CrNiCu14_5.dll”

Names of Measured Data (*_meas.txt)

1. Should have the same name as the recall functions (*.dll)
2. Should, additionally, end in “_meas”, standing for “measured” data

e.g.: “nC15_meas.txt”, “nX10Cr13_meas.txt”

Contents of Measured Data (*_meas.txt)

1. The first line should provide the parameter names (Phip, Phi, Temp, Kf)
2. The names should be exactly these – “Phip”, “Phi”, “Temp” and “Kf”
3. All values and names should be TAB separated
4. Each case should be on a new line
5. Should be represented in standard format (not in scientific representation)

e.g.:

PHIP	PHI	TEMP	KF
0,1	0,04	20	328,16
0,1	0,06	20	373,91
0,1	0,08	20	413,21

Names of Regression Coefficients (*_regr.txt)

1. Should have the same name as the recall functions (*.dll)
2. Should, additionally, end in “_regr”, standing for “regression” coefficients

e.g.: “nC15_regr.txt”, “nX10Cr13_regr.txt”

Contents of Regression Coefficients (*_regr.txt)

1. Each coefficient should be on a separate line
2. There should be exactly six coefficients in order M_0 (the first) through M_5 (the last)
3. Should be represented in standard format (not in scientific representation)

e.g.:

```
2707,70769192214
-0,00288377612389429
-0,0205573178623678
0,259427166935543
-0,355751343984173
0,000168955205886425
```

Location of files (*.dll, *.txt)

1. Recall functions (*.dll) should be placed in the subfolder “dlls” in the main folder of the software
2. Measured data (*_meas.txt) should be placed in the subfolder “measured” in the main folder of the software
3. Regression coefficients (*_regr.txt) should be placed in the subfolder “regression coefficients” in the main folder of the software