

**VariReg ver. 0.10.2**

**A SOFTWARE TOOL FOR REGRESSION MODELLING  
USING VARIOUS MODELLING METHODS**

Gints Jekabsons

Institute of Applied Computer Systems  
Riga Technical University  
Meza 1/3, LV-1048, Riga, Latvia  
E-mail: [gints.jekabsons@rtu.lv](mailto:gints.jekabsons@rtu.lv)  
URL: <http://www.cs.rtu.lv/jekabsons/>

**User's Manual**

May, 2010

# CONTENTS

1. INTRODUCTION .....	3
2. IMPLEMENTED REGRESSION MODELLING METHODS .....	5
2.1. Error measures .....	5
2.2. “Full” polynomials .....	6
2.3. Sparse polynomials and subset selection .....	6
2.4. Adaptive Basis Function Construction .....	8
2.5. Locally Weighted Polynomials .....	9
2.6. k-Nearest Neighbours .....	11
2.7. Radial Basis Function interpolation .....	11
2.8. Kriging interpolation .....	12
2.9. Multivariate Adaptive Regression Splines .....	13
2.10. Polynomial Neural Networks .....	13
2.11. Model averaging/ensembling/combining .....	14
3. IMPLEMENTED OPTIMIZATION ALGORITHMS .....	16
3.1. Particle Swarm Optimization .....	16
3.2. A simple Grid Search .....	17
4. VARIREG USER’S INTERFACE .....	18
4.1. Input file format .....	18
4.2. User’s interface .....	18
4.2.1. Tab “Data” .....	19
4.2.2. Tab “Polynomials” .....	20
4.2.3. Tab “ABFC” .....	22
4.2.4. Tab “LWP” .....	23
4.2.5. Tab “k-NN” .....	24
4.2.6. Tab “RBF” .....	25
4.2.7. Tab “Kriging” .....	26
4.2.8. Tab “MARS” .....	27
4.2.9. Tab “PNN” .....	28
4.2.10. Tab “Averaging” .....	29
4.2.11. Tab “Optimization” .....	30
5. USING VARIREG FROM COMMAND LINE AND FROM MATLAB .....	32
5.1. Using VariReg from command line .....	32
5.2. Using VariReg from Matlab .....	33
6. REFERENCES .....	34

# 1. INTRODUCTION

VariReg is a software tool for general purpose multidimensional regression modelling with the main emphasis on methods used in metamodelling / surrogate modelling. VariReg was originally developed for experimental and educational purposes and it is primarily intended for use on small and moderately-sized numerical data sets. The tool provides means for creating “full” polynomial regression models (also called Response Surface models), sparse polynomial models (also called partial polynomial models) employing subset selection algorithms, such as Sequential Forward Selection (SFS; also known as Forward Selection or Forward Stepwise Selection), Steepest Descent Hill Climbing (SDHC), Random Restart Hill Climbing (RRHC), and Sequential Floating Forward Selection (SFFS), as well as different other regression modelling techniques – Adaptive Basis Function Construction (ABFC), Locally Weighted Polynomials (LWP), k-Nearest Neighbours (k-NN), Radial Basis Function (RBF) interpolation, Kriging interpolation, Multivariate Adaptive Regression Splines (MARS), and Polynomial Neural Networks (PNN) induced by Group Method of Data Handling (GMDH). In the methods, the model evaluation and hyperparameter selection is done using one of the following criteria: F-test, Corrected Akaike’s Information Criterion (AICC), Schwarz’s Bayesian Information Criterion (BIC), Generalized Cross-Validation (GCV),  $\nu$ -fold Cross-Validation (CV), Leave-One-Out Cross-Validation (LOOCV), or a simple Hold-Out. The modelling methods (or models generated by them) can be evaluated using  $\nu$ -fold Cross-Validation or Hold-Out. The full and sparse polynomial models can also be represented in a "spreadsheet-friendly" way.

VariReg also enables all the implemented regression modelling methods to be combined together using a number of model averaging techniques: simple unweighted averaging, averaging weighted by LOOCV error, LOOCV error variance, or LOOCV error correlation, as well as averaging by Stacking.

All the VariReg’s regression modelling methods, including model averaging, can also be put to work from:

- a command line using .ini configuration files;
- Matlab environment using wrapper functions provided together with VariReg.

Note that the implementations of regression modelling methods in VariReg can be considerably faster (even orders of magnitude) than the same methods scripted in Matlab.

In the context of metamodelling / surrogate modelling, VariReg can be employed for building metamodels / surrogate models for evaluation and comparison of the different techniques as well as for further use in what-if analysis, design optimization, design space exploration etc.

VariReg also provides means for optimization of the values of output variables using the built regression models as objective functions. In the current version of VariReg the following optimization algorithms are implemented: Particle Swarm Optimization (PSO) and a simple Grid Search (GS).

This user’s manual provides a brief overview of the implemented regression modelling methods as well a short guide to VariReg’s input file format, user’s interface, and provided access from command line and Matlab environment.

Note that the current version of VariReg is implemented as single-threaded and the software may appear as hung-up while performing some longer model building, prediction, or optimization operations.

The VariReg tool can be downloaded at <http://www.cs.rtu.lv/jekabsons/>.

## Citations

If you are using VariReg software and/or ABFC methods for your scientific research, please give at least one of the following references:

- **VariReg software:**
  - Jekabsons G., VariReg: A software tool for regression modelling using various modelling methods, 2010, available at <http://www.cs.rtu.lv/jekabsons/>
  - reference (Jekabsons 2010)
  
- **F-ABFC method:**
  - reference (Jekabsons 2010)
  - reference (Jekabsons & Lavendels 2008)
  
- **EF-ABFC method:**
  - reference (Jekabsons 2010)
  - reference (Jekabsons 2008)
  
- **The particular LWP implementation:**
  - reference (Kalnins et al. 2008)

## 2. IMPLEMENTED REGRESSION MODELLING METHODS

A regression model describes a relation between a vector of  $d$  real valued input variables (features)  $x = (x_1, x_2, \dots, x_d)$  and a single real valued output variable  $y$ . Using a finite number  $n$  of training observations (data cases or data points)  $(x_{(i)}, y_{(i)})$ ,  $i = 1, 2, \dots, n$  one wants to build a model  $F$  that allows predicting the output value for yet unseen input values as closely as possible (Hastie 2009, Cherkassky & Mulier 2007).

### 2.1. Error measures

See Table 1 for how the error measures used in VariReg are calculated, where  $\bar{y}$  is the mean of the observed values of  $y$ . The lower the value of an error measure (except STD and VAR), the closer the model is to the particular data points. In VariReg the main results are displayed in RRMSE form. It is calculated by dividing RMSE by STD. While RMSE represents model's deviation from the data, the STD captures how irregular the problem is. With validation or test data sets, RRMSE equal to 0 is a perfect fit; RRMSE value approaching 1 indicates that the model is no better than a mean value of  $y$ ; RRMSE value largely exceeding 1 indicates that the model probably drastically overfits the training data. For  $R^2$  the opposite is true.

All the measures are used for training data (calculation of training error for calculation of values of complexity penalization criteria), for validation data (calculation of validation error with Hold-Out or Cross-Validation), as well as for test data (calculation of test error with Hold-Out or Cross-Validation). With Cross-Validation the measures are calculated separately for each Cross-Validation fold and in the end an averaged value is calculated.

Table 1

Calculation of error measures (here  $n$  denotes the number of data points in the used set – training, validation, or test set)

Measure	Calculation
Sum of Squared Error (SSE)	$SSE = \sum_{i=1}^n (y_{(i)} - F(x_{(i)}))^2$
Mean Squared Error (MSE)	$MSE = \frac{SSE}{n} = \frac{1}{n} \sum_{i=1}^n (y_{(i)} - F(x_{(i)}))^2$
Root Mean Squared Error (RMSE)	$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_{(i)} - F(x_{(i)}))^2}$
Standard Deviation (STD)	$STD = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_{(i)} - \bar{y})^2}$
Relative Root Mean Squared Error (RRMSE)	$RRMSE = \frac{RMSE}{STD} = \frac{\sqrt{\frac{1}{n} \sum_{i=1}^n (y_{(i)} - F(x_{(i)}))^2}}{\sqrt{\frac{1}{n} \sum_{i=1}^n (y_{(i)} - \bar{y})^2}}$
Variance (VAR)	$VAR = \frac{1}{n} \sum_{i=1}^n (y_{(i)} - \bar{y})^2$
$R^2$	$R^2 = 1 - \frac{MSE}{VAR} = 1 - \frac{\frac{1}{n} \sum_{i=1}^n (y_{(i)} - F(x_{(i)}))^2}{\frac{1}{n} \sum_{i=1}^n (y_{(i)} - \bar{y})^2}$

## 2.2. “Full” polynomials

Low-degree “full” polynomials are the most widely used regression models. For example, first-degree full polynomial can be defined as

$$F(x) = a_0 + \sum_{i=1}^d a_i x_i \quad (1)$$

and second-degree full polynomial can be defined as

$$F(x) = a_0 + \sum_{i=1}^d a_i x_i + \sum_{i=1}^d \sum_{j=i}^d a_{ij} x_i x_j \quad (2)$$

where  $a_0$ ,  $a_i$ ,  $a_{ij}$ , are model’s parameters and  $d$  is the number of input variables. VariReg tool implements full polynomials of any user-predefined degree in interval  $[0,100]$ .

Generally, a polynomial regression model may also be defined as a linear summation of basis functions:

$$F(x) = \sum_{i=1}^k a_i f_i(x) \quad (3)$$

where  $k$  is the number of basis functions (equal to the number of model’s parameters);  $f_i(x)$  ( $i=1,2,\dots,k$ ) is a predefined polynomial basis function. The number of the basis functions in polynomial model of degree  $p$  is

$$m = \prod_{i=1}^p (1 + d / i). \quad (4)$$

The estimation of model’s parameters is made based on the training data typically using the Ordinary Least-Squares (OLS) method, minimizing

$$a = \arg \min_a \sum_{i=1}^n (y_{(i)} - F(x_{(i)}))^2 \quad (5)$$

where  $x_{(i)}$  is the vector of input variables’ values of the  $i$ th data point and  $y_{(i)}$  is the output value of that point. In VariReg, for “full” polynomials as well as all the other regression modelling methods discussed below the systems of linear equations in OLS are solved using Gaussian elimination and backsubstitution.

## 2.3. Sparse polynomials and subset selection

Low-degree polynomial approximations have been well accepted in practice, as they require low number of data points and are computationally very efficient. On other hand they can not approximate highly nonlinear behaviours. Instead, higher-degree polynomials can be employed. However, if no special care is taken, they tend to overfit the data and produce high errors in regions where the data points are relatively sparse. One possible remedy for the overfitting problem is employment of the subset selection methods. The methods are aimed to identify the best subset of polynomial terms (or basis functions) to include in the model and to remove the unnecessary ones, in this manner creating a sparse polynomial model of increased predictive performance. VariReg tool implements methods for building sparse polynomial models of any user-predefined degree in interval  $[0,100]$ .

In the subset selection, if the full set of predefined basis functions is  $\phi$  then any of its subsets  $f \subseteq \phi$  can be specified as a binary vector  $s$  with cardinality  $m = |s|$  (i.e. the total number of predefined basis functions) in which the  $j$ th value equal to 1 means that the  $j$ th basis function  $\phi_j$  is included in the subset forming the model but a value equal to 0 means that it is not:

$$f = \{\phi_j \mid j = 1, 2, \dots, m; s_j = 1\}. \quad (6)$$

Usually, the full set of basis functions  $\phi$  is equal to the set of basis functions in a full polynomial model of a user-predefined maximal degree  $p$  (a non-negative integer). Hence, for example, if  $d = 3$  and  $p = 2$ , then the full (ordered) set contains  $m = 10$  basis functions

$$\phi = \{1, x_1, x_2, x_3, x_1^2, x_2^2, x_3^2, x_1x_2, x_1x_3, x_2x_3\}$$

and the vector

$$s = (1, 0, 1, 0, 1, 1, 0, 0, 1, 0)^T$$

corresponds to the subset

$$f = \{1, x_2, x_1^2, x_2^2, x_1x_3\}$$

which in turn corresponds to the model

$$F(x) = a_1 + a_2x_2 + a_3x_1^2 + a_4x_2^2 + a_5x_1x_3.$$

Formally, the problem of finding the best subset  $f^*$  can be defined as finding the best combination of values for  $m$  bits in the vector  $s$ :

$$s^* = \arg \min_s J(\{\phi_j \mid j = 1, 2, \dots, m; s_j = 1\}) \quad (7)$$

where  $J(\cdot)$  is a criterion which evaluates the predictive performance of the regression model resulting from the subset.

Searching through all the possible combinations of  $m$  bits requires evaluation of  $2^m$  models, i.e. exponential runtime and thus is impractical in most cases. Hence in subset selection heuristic search algorithms are used. They efficiently traverse the space of subsets, by adding and deleting basis functions and use an evaluation measure that directs the search into areas of increased performance, i.e. where the “better” models are.

VariReg implements the following heuristic search algorithms for subset selection (Pudil et al. 1995, Hand et al. 2001, Miller 2002, Russell & Norvig 2002, Webb 2002):

- Sequential Forward Selection (SFS; also known as Forward Selection);
- Steepest Descent Hill Climbing (SDHC) with optionally randomized starting model (transforming the algorithm to the Random Restart Hill Climbing, RRHC);
- Sequential Floating Forward Selection (SFFS).

All the algorithms can be used together with the following criteria (except that SFS can also be used together with the F-test):

- Small sample corrected Akaike’s Information Criterion (AICC) (Hurvich & Tsai 1989);
- Bayesian Information Criterion (BIC) (Schwarz 1978) – equal to the two-part Minimum Description Length criterion (Rissanen 1978);
- Generalized Cross-Validation (GCV) (Golub et al. 1979, Craven & Wahba 1979);
- $\nu$ -fold Cross-Validation (CV) (Zhang 1993, Kohavi 1995, Hastie 2009).

Table 2 shows how the values of the complexity penalization criteria are calculated.

Table 2

Calculation of the complexity penalization criteria

Criterion	Equation
AICC	$AICC = n \ln(MSE) + 2k + (2k(k + 1))/(n - k - 1)$
BIC	$BIC = n \ln(MSE) + k \ln(n)$
GCV	$GCV = n \ln(MSE) - 2n \ln(1 - k/n)$

SFS is one of the simplest and most widely known search algorithms for subset selection. It starts with the simplest model (typically an empty subset of basis functions or a subset of only the intercept term in it) and iteratively adds the functions leading to the highest performance increase (according to the chosen criterion – AICC, BIC, GCV, or CV) until the model performance cannot be enhanced any further by adding a single basis function.

SFS with F-test also starts with the simplest model and sequentially adds those basis functions to the model that most significantly improves its fit to the training data. To avoid overfitting, the significance of the MSE improvement gained by adding the basis function to the current model is tested. The improvement is significant, if the obtained  $F$  value

$$F = \frac{(MSE_H - MSE_M)/(k_M - k_H)}{MSE_M/(n - k_M)} \quad (8)$$

is greater than a predefined significance threshold value  $F_\alpha(k_M - k_H, n - k_M)$  where  $k_M$  is the number of basis functions in the full model (with all the basis function included);  $k_H$  is the number of basis functions in the current model;  $MSE_M$  is the MSE of the full model;  $MSE_H$  is the MSE of the current model;  $\alpha = 0.15$  (a recommended value for subset selection (Hocking 2003)). SFS proceeds with greedy search by choosing the largest significant improvement and stops, if no significant improvement is possible.

VariReg also implements a possibility to automatically select the best maximal degree using CV however this process requires significantly larger computational resources.

## 2.4. Adaptive Basis Function Construction

The approach of subset selection assumes that the chosen fixed full set of predefined basis functions (in the VariReg it is chosen in the most common way – by setting the maximal degree of the polynomial) contains a subset that is sufficient to describe the target relation sufficiently well. Hence, the efficiency of subset selection largely depends on whether or not the predefined set of basis functions contains such a subset.

There exists a different approach for polynomial model building which does not assume a predefined set of basis functions – Adaptive Basis Function Construction (Jekabsons 2010, Jekabsons & Lavendels 2008, Jekabsons 2008). The approach allows generating polynomials of arbitrary complexity and degree without the requirement to predefine any basis functions or to set the maximal degree of the polynomial – all the required basis functions are constructed adaptively specifically for the data at hand. At the same time the required computational resources are considerably reduced comparing to those required for performing subset selection (Jekabsons 2010, Jekabsons & Lavendels 2008, Jekabsons 2008).

As already said, generally, a polynomial model can be defined by a linear summation of basis functions:

$$F(x) = \sum_{i=1}^k a_i f_i(x) \quad (9)$$

where the coefficients  $a$  are calculated using OLS;  $f_i(x)$  is a basis function which generally can be defined as a product of original input variables each with an individual exponent:

$$f_i(x) = \prod_{j=1}^d x_j^{r_{ij}} \quad (10)$$

where  $r$  is a  $k \times d$  matrix of non-negative integer exponents such that  $r_{ij}$  is the exponent of the  $j$ th variable in the  $i$ th basis function. Note that when for a particular  $i$ th basis function  $r_{ij} = 0$  for all  $j$ , the basis function is the intercept term.

Given a number of input variables  $d$ , matrix  $r$  with a specified number of rows  $k$  and with specified values of each of its element completely defines a model with all its basis functions. The set of basis functions included in a model is then equal to

$$f = \left\{ \prod_{j=1}^d x_j^{r_{i,j}} \mid i = 1, 2, \dots, k \right\}. \quad (11)$$

For example, if  $d = 3$  and  $k = 4$ , then the matrix

$$r = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 3 \\ 1 & 1 & 1 \end{bmatrix}$$

corresponds to the set

$$f = \{x_1^0 x_2^0 x_3^0, x_1^1 x_2^0 x_3^0, x_1^0 x_2^1 x_3^3, x_1^1 x_2^1 x_3^1\} = \{1, x_1, x_2 x_3^3, x_1 x_2 x_3\}$$

which in turn corresponds to the model

$$\begin{aligned} F(x) &= a_1 x_1^0 x_2^0 x_3^0 + a_2 x_1^1 x_2^0 x_3^0 + a_3 x_1^0 x_2^1 x_3^3 + a_4 x_1^1 x_2^1 x_3^1 = \\ &= a_1 + a_2 x_1 + a_3 x_2 x_3^3 + a_4 x_1 x_2 x_3. \end{aligned}$$

Formally, the problem of finding the best subset  $f^*$  can be defined as finding the best combination of values for  $m$  bits in the matrix  $r$ :

$$r^* = \arg \min_r J \left( \left\{ \prod_{j=1}^d x_j^{r_{i,j}} \mid i = 1, 2, \dots, k \right\} \right). \quad (12)$$

As neither the upper bounds of  $r$  elements' values nor the upper bound of  $k$  are defined, it is possible to generate polynomial models of arbitrary complexity, i.e. of arbitrary number of basis functions each with an arbitrary exponent for each of the input variables. This makes the model building process highly flexible.

Construction of the model is carried out in an iterative manner directly with the matrix  $r$  using a number of simple so-called model refinement operators that allow adding, copying, modifying, and deleting the rows of  $r$ , i.e. adding, copying, modifying, and deleting the basis functions of the model. The employed search (combinatorial optimization) procedure is a modification of the SFFS algorithm (hence the method is called Floating ABFC, i.e. F-ABFC) while models are evaluated using the AICC (in VariReg also BIC and GCV can be employed here). A more detailed description of the F-ABFC is given in (Jekabsons 2010, Jekabsons & Lavendels 2008).

However, there are two issues that may plague all the methods of model building (including subset selection and ABFC), especially when working with relatively small data sets: the selection bias and the selection instability (also called selection variance) (Jekabsons 2010, Jekabsons 2008). In order to deal with the two issues, a method of ensembling the models built by F-ABFC is proposed (Jekabsons 2010, Jekabsons 2008) – Ensemble of Floating Adaptive Basis Function Construction (EF-ABFC). It is done using a collaboration of two techniques: 1) CV performed over the entire F-ABFC search process for selection of one final best model from the best models of each F-ABFC iteration (the validation set is not used for model evaluation during the search, instead it is used strictly only for evaluation and selection after the main search process has already ended). This additional evaluation can detect whether the search process at some iteration might have started to generate overfitted models and select a model of some earlier iteration that is hopefully not (or at least less) overfitted; 2) the  $\nu$  models from the  $\nu$  CV loops are combined using a simple ensemble technique – unweighted model averaging. Combining models in this way has the effect of smoothing out erratic models that overfit the data and gain more stability in the modelling process (especially for small and noisy data). However, there is also a disadvantage for the EF-ABFC – it requires larger computational resources.

## 2.5. Locally Weighted Polynomials

Locally Weighted Polynomial approximation (also called Locally Weighted Regression or Moving Least Squares) (Cleveland & Devlin 1988) is designed to address situations in which the models of global behaviour do not perform well or cannot be effectively applied without undue effort. The LWP approximation is carried out by pointwise fitting of low-degree polynomials to

localized subsets of the data. The advantage of this method is that the analyst is not required to specify a global function of the data. However, the method requires considerably higher computational resources for finding the best values of the hyperparameters as well as when LWP is finally used for prediction.

The assumption of the LWP approximation is that near the query point the value of the actual response changes smoothly and can be approximated using a low-degree polynomial. The coefficients of the polynomial are calculated using the Weighted Least-Squares method giving the largest weights to the nearest (usually according to the Euclidean distance) data points and the smallest or zero weights to the farthest data points.

The coefficients  $a$  of the polynomial are calculated, minimizing

$$a = \arg \min_a \sum_{i=1}^n w(x_{query}, x_{(i)}) (F(x_{(i)}) - y_{(i)})^2 \quad (13)$$

where  $w$  is a weight function;  $x_{query}$  is the query point nearest neighbours of which will get the largest weights. The weight function  $w$  depends on the Euclidean distance (in scaled space) between the point of interest  $x_{query}$  and the points of observations  $x$ . One of the most widely used weight functions is the Gaussian weight function. In VariReg there are two different kinds of weighting for Gaussian weight function implemented.

The first kind uses a weight function which takes into account the distance to the farthest point. It is defined as follows:

$$w(x_{query}, x_{(i)}) = \exp(-\alpha \mu_{(i)}^2) \quad (14)$$

where  $\alpha$  is a coefficient and the  $\mu_{(i)}$  is a scaled distance from the query point to the  $i$ th point in the training data set:

$$\mu_{(i)} = \frac{\|x_{query} - x_{(i)}\|}{\|x_{query} - x_{farthest}\|} \quad (15)$$

where  $\|\cdot\|$  is the Euclidean norm;  $x_{farthest}$  is the farthest training point from the point  $x_{query}$ . The locality of the approximation is controlled by varying the value of the coefficient  $\alpha$ . If  $\alpha$  is equal to zero then local approximation transforms into global approximation.

The second kind of weighting uses a weight function which does not take into account the distance to the farthest point. It is defined as follows:

$$w(x_{query}, x_{(i)}) = \exp\left(-\frac{\mu_{(i)}^2}{2\alpha^2}\right) \quad (16)$$

where  $\alpha$  is a coefficient and  $\mu_{(i)}$  is the distance from the query point to the  $i$ th point in the training data set:

$$\mu_{(i)} = \|x_{query} - x_{(i)}\|. \quad (17)$$

For this kind of weighting, the locality of the approximation is also controlled by varying the value of the coefficient  $\alpha$ . However, the lower is the  $\alpha$ , the more local is the approximation.

In VariReg, for both these kinds of weighting, the best value of  $\alpha$  can be found using the LOOCV technique. It is done using the following algorithm. First, a minimum of LOOCV is found using larger steps (an increment of 10 or 0.1 for each of the two kinds correspondingly) from a starting value (0 or 0.001 correspondingly). This search is stopped when there were already 5 steps without LOOCV improvement. After that the second phase is engaged where a refined search with smaller steps is performed in the region of the minimum until the step size is lower than 0.025 or 2 correspondingly.

VariReg enables working with LWP of user-predefined degree or choose the best degree automatically using LOOCV.

## 2.6. k-Nearest Neighbours

Generally, the formula for k-NN can be written as

$$F(x) = \frac{\sum_{i=1}^k w_i y'_{(i)}}{\sum_{i=1}^k w_i} \quad (18)$$

where  $y'_{(i)}$  is the output value of  $i$ th point from the  $k$  points nearest to the query point  $x$  and  $w_i$  is the weight of that point. Here closeness implies the Euclidean distance.

VariReg implementation of k-NN offers three distance weighting schemes (Witten & Frank 2005):

- no weighting, i.e.  $w_i = 1$  ( $i = 1, 2, \dots, k$ );
- $w_i = 1/\|x - x'_{(i)}\|$  ( $i = 1, 2, \dots, k$ );
- $w_i = 1 - \|x - x'_{(i)}\|$  ( $i = 1, 2, \dots, k$ ),

where  $x'_{(i)}$  is the input value of  $i$ th point from the nearest ones. For the second scheme, when a distance is equal to zero its weight is assumed to be infinite – a simple average of all the  $y'_{(i)}$ 's for which the distance is zero is calculated.

## 2.7. Radial Basis Function interpolation

RBF was originally developed for scattered multivariate data interpolation (Hardy 1971). It uses a series of basis functions that are symmetric and centred at each sampling point. Radial basis functions are a special class of functions with their main feature being that their response decreases (or increases) monotonically with distance from a central point (Queipo et al. 2005). The centre, the distance scale, and the precise shape of the radial function are parameters of the model. The RBF interpolation implemented in VariReg can be expressed as follows (Jin et al. 2002):

$$F(x) = \mu + \sum_{i=1}^n a_i \phi(\|x - x_{(i)}\|) \quad (19)$$

where  $\mu$  is either a polynomial model or a constant value;  $a$  are coefficients calculated by solving a linear system;  $\phi$  is a basis function, for which there are many different choices, such as multiquadric, inverse multiquadric, polyharmonic, thin plate spline, Gaussian, etc. Detailed review on RBF methods can be found in (Powell 1987, Gutmann 2001). In VariReg three of the most popular and efficient (e.g. see (McDonald et al. 2000, Colaco & Dulikravich 2008)) types of basis functions are implemented: multiquadric, thin plate spline, and Gaussian. Multiquadric basis functions are defined as follows

$$\phi(dist) = \sqrt{dist^2 + c^2} \quad (20)$$

and thin plate spline basis functions are defined as follows

$$\phi(dist) = (dist^2 + c^2) \ln(\sqrt{dist^2 + c^2}) \quad (21)$$

where the shape parameter  $c$  is kept constant, e.g. as 1 (Acar & Rais-Rohani 2009) or as  $1/n$  (Colaco et al. 2007). And the  $\mu$  is defined as a constant value – mean of  $y$ :

$$\mu = \frac{1}{n} \sum_{i=1}^n y_i \quad (22)$$

Gaussian basis functions are defined as follows

$$\phi(dist) = \exp\left(-\frac{dist^2}{2\delta^2}\right) \quad (23)$$

where  $\delta$  is the radius (also called smoothing parameter) of the basis function which in VariReg may be set manually by the user or can be found automatically using LOOCV. The automatic search is done by trying different values starting from 0.1 and adding 0.1 (if current  $\delta$  is lower than 1.0) or adding 1.0 (if current  $\delta$  is higher than or equal to 1.0). The search is stopped when in series of five steps no improvement over the so-far-best  $\delta$  value was found.

A special case of RBF interpolation is the Shepard interpolation (Shepard 1968, Press et. al. 2007). In VariReg, the simplest form of Shepard interpolation is implemented. The interpolation formula is

$$F(x) = \frac{\sum_{i=1}^n y_{(i)} \phi(\|x - x_{(i)}\|)}{\sum_{i=1}^n \phi(\|x - x_{(i)}\|)} \quad (24)$$

(with appropriate provision for the limiting case where  $x$  is equal to one of the  $x_{(i)}$ 's). Note that no solution of linear equations is required.

Shepard proposed the simple power-law function

$$\phi(\text{dist}) = \text{dist}^{-c} \quad (25)$$

with (typically)  $1 < c < 3$ . While the Shepard interpolation is rarely as accurate as other RBF interpolations, it is simple and computationally very efficient.

## 2.8. Kriging interpolation

Kriging interpolation (sometimes also called Spatial Correlation Modelling or even Gaussian Process Regression) was originally developed by Georges Matheron based on the work of Daniel Gerhardus Krige. The mathematical form of a Kriging model has two parts

$$F(x) = \mu + Z(x) \quad (26)$$

where  $\mu$  represents a global model (in VariReg it is a polynomial model of user-defined degree 0, 1, or 2) and  $Z(x)$  is the realization of a stochastic process with mean zero and spatial correlation function given by

$$\text{Cov}(Z(x_{(i)}), Z(x_{(j)})) = \sigma^2 R(x_{(i)}, x_{(j)}) \quad (27)$$

where  $\sigma^2$  is the process variance and  $R$  is a correlation function (Jin et al. 2001, Jin et al. 2002, Martin & Simpson 2005). A variety of correlation functions can be chosen (Simpson et al. 1998). VariReg offers exponential, Gaussian, linear, spherical, and cubic spline correlation functions.

The choice of correlation function should be motivated by the underlying phenomenon, e.g. a function to be optimized or a physical process to be modelled. If the underlying phenomenon is continuously differentiable, the correlation function will likely show a parabolic behaviour near the origin, which means that the Gaussian or the spline function should be chosen. Conversely, physical phenomena usually show a linear behaviour near the origin, and exponential, linear, or spherical would usually perform better (Lophaven et al. 2002, Isaaks & Srivastava 1989).

The advantage of the Kriging method is that it is very flexible. The major disadvantage of the method is that the model construction can be very time-consuming as determining of the correlation function parameters used to fit the model is a  $d$ -dimensional optimization problem. Moreover, the correlation matrix can become singular if multiple data points are spaced close to one another or if the data points are generated from particular designs (Jin et al. 2001).

In the VariReg, the user must provide initial guess, as well as lower bound and upper bound of correlation function parameters.

Kriging implementation in VariReg is based on the DACE (Design and Analysis of Computer Experiments) Toolbox (Lophaven et al. 2002) (<http://www2.imm.dtu.dk/~hbn/dace/>) and requires Matlab to be installed on the computer. In VariReg the user must only browse for the `matlab.exe` file – no additional configuration of Matlab or special installation of the DACE toolbox is required.

## 2.9. Multivariate Adaptive Regression Splines

Multivariate Adaptive Regression Splines was originally developed by Jerome H. Friedman (Friedman 1991a, Friedman 1991b, Friedman 1993) as a method for flexible regression modelling of high dimensional data (i.e. a large number of input variables). The model takes the form of an expansion in product spline basis functions, where the number of basis functions as well as the parameters associated with each one (product degree and knot locations) are automatically determined by the data through a forward/backward iterative approach.

MARS model can be defined as a sum of basis functions:

$$F(x) = a_0 + \sum_{i=1}^k a_i f_i(x) \quad (28)$$

where  $f_0(x)$  is a basis function;  $k$  is the number of basis functions in the model except for the constant basis function  $f_0(x) = 1$  coefficient of which is the  $a_0$ . All the coefficients  $a$  are calculated using the OLS. The basis functions are of the form

$$f_i(x) = \prod_{j=1}^{d_i} [s_{ji} (x_{v(j,i)} - t_{ji})]_+ \quad (29)$$

where  $d_i$  is the number of variables (interaction order) in the  $i$ th basis function;  $s_{ji} = \pm 1$ ;  $x_{v(j,i)}$  is the  $v$ th variable,  $1 \leq v(j,i) \leq d$ ;  $t_{ji}$  is knot location on each of the corresponding variables. The subscript “+” means that the function is a truncated power function.

The user must adjust the following options:

- Maximum number of basis functions;
- Maximum number of interactions;
- Speed (1 – no acceleration; 5 – maximum speed advantage; default is 4);
- Penalty per knot (1..9; default is 3; the best value can also be found using 10-fold Cross-Validation). Theory suggests values in the range of about 2 to 4. Larger values will lead to fewer knots being placed. The recommended (and default) value is 3 (Friedman 1991b);
- Penalty for increasing the number of variables in the MARS model, sometimes useful with highly collinear designs as it may produce nearly equivalent models with fewer input variables (0 – no penalty; 0.05 – moderate penalty; 0.1 – heavy penalty; default is 0);
- Interpolation type (piecewise-linear or piecewise-cubic).

VariReg implements MARS version 3.6 based on the original Fortran source code of Friedman which can be found for example in XTAL software package (<http://www.ece.umn.edu/users/cherkass/ee8591/software/xtal.html>).

## 2.10. Polynomial Neural Networks

The GMDH method is mainly referred as a method for self-organizing polynomial neural networks (Ivakhnenko 1968, Barron et al. 1984, Farlow 1984, Elder IV & Brown 2000). The most widely known of its variations work exclusively with polynomials and therefore also the result of the method can be written in polynomial form.

The building blocks of GMDH usually are second or third degree polynomials (often they are sparse polynomials generated by some algorithm of subset selection) of two or three input variables. Such building blocks, also called partial descriptions (PDs), like neurons in neural networks, are arranged in layers (see Figure 1). The coefficients of each PD are calculated using the OLS by trying to approximate the original dependent variable  $y$  of training data. The exact number of layers and connections of the network as well as the structure of PDs is not set a priori but is the object of search layer by layer. The maximal number of PDs selected in each layer is usually set equal to the number of original input variables.

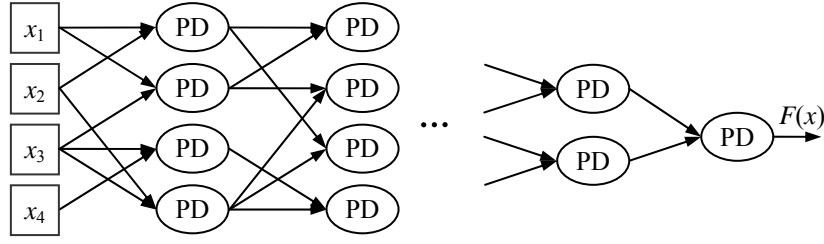


Figure 1. An example of GMDH polynomial neural network structure

GMDH is argued to be advantageous in handling a relatively big number of variables (including irrelevant ones) and noisy or small data samples (Ivakhnenko 1968, Elder IV & Brown 2000). In comparison with one-step-greedy methods, GMDH grows models by blocks of terms at a time.

There exists a wide spectrum of GMDH methods, even if only those which work exclusively with polynomial PDs are considered (e.g. see (Ivakhnenko 1968, Barron et al. 1984, Farlow 1984, Elder IV & Brown 2000, Anastasakis & Mort 2001)). VariReg implements a variation of GMDH which is similar to the classical combinatorial algorithm (Ivakhnenko 1968, Farlow 1984).

The user must adjust the following settings:

- maximum degree for polynomials in PDs;
- maximum number of inputs for PDs (2 or 3);
- whether the PDs are full polynomials or their structure is generated by a subset selection algorithm (SFS, SDHC, or SFFS);
- criterion for subset selection and stopping of network building (note that in VariReg the complexity of the network is estimated by calculating the sum of the number of parameters in all the PDs which are directly or indirectly connected to the best PD in the last layer);
- the maximum number of PDs in each layer;
- whether in each successive layer PDs take inputs from PDs of immediately preceding layer or also from the original input variables.

## 2.11. Model averaging/ensembling/combining

Model combination approaches are an attempt to take advantage of multiple models instead of just a single model in the hope that the models would correct each others errors.

Typical model combination procedures consist of a two-stage process (Cherkassky & Mulier 2007). In the first stage, the training data are used to separately estimate a number of different models. The parameters of these models are then held fixed. In the second stage, these individual models are linearly combined to produce the final model.

The combined model can be expressed as follows:

$$F_{comb}(x) = \sum_{i=1}^b w_i F_i(x) \quad (30)$$

where  $F_{comb}(x)$  is the combined model;  $b$  is the number of models combined;  $w_i$  is the weight for the  $i$ th model;  $F_i(x)$  is the  $i$ th model.

VariReg implements a number of procedures of combining the models:

- a simple unweighted average;
- averaging weighted by LOOCV error;
- averaging weighted by LOOCV error variance;
- averaging weighted by LOOCV error correlation;
- averaging by Stacking employing OLS;
- averaging by Stacking employing Non-negativity constrained Least Squares (NNLS).

For the unweighted average, the weights are computed as:

$$w_i = 1/b. \quad (31)$$

The unweighted averaging does not take into account that the different models may not have the same level of accuracy. The other implemented model combining procedures on the other hand in general try to select the weights such that the models with high accuracy have larger weights than those of models with lower accuracy hence attempting to enhance the predictive performance of the whole ensemble. For these procedures, the calculation of weights depends on the LOOCV resampling technique where the performance of an individual model is estimated by building the model  $n$  times while each time keeping out of the training data a different data point and predicting its output value using the built model (Cherkassky & Mulier 2007, Acar & Rais-Rohani 2009).

The averaging based on the LOOCV error calculates an SSE value for each of the models:

$$SSE_i = \sum_{j=1}^n (y_{(j)} - F_i^{-j}(x_{(j)}))^2 \quad (32)$$

where  $F_i^{-j}(x)$  is the  $i$ th model built on the training data with the  $j$ th data point excluded. Then the weights are computed as:

$$w_i = \frac{1}{SSE_i} \bigg/ \sum_{j=1}^b \frac{1}{SSE_j}. \quad (33)$$

Here the division by the sum of inverse errors ensures that the following requirement is satisfied:

$$\sum_{i=1}^b w_i = 1. \quad (34)$$

The averaging based on the LOOCV error variance computes the weights as (Acar & Rais-Rohani 2009):

$$w_i = \frac{1}{VAR_i} \bigg/ \sum_{j=1}^b \frac{1}{VAR_j}, \quad VAR_i = \sum_{j=1}^n \left( \frac{SSE_i}{n} - (y_{(j)} - F_i^{-j}(x_{(j)}))^2 \right)^2. \quad (35)$$

The averaging based on the LOOCV error correlation computes the weights as (Bishop 1995, Acar & Rais-Rohani 2009):

$$w_i = \sum_{j=1}^b (\mathbf{C}^{-1})_{ij} \bigg/ \sum_{l=1}^b \sum_{j=1}^b (\mathbf{C}^{-1})_{lj}. \quad (36)$$

where  $\mathbf{C}$  is the error correlation matrix whose elements are calculated from

$$\mathbf{C}_{ij} = \frac{1}{n} \sum_{l=1}^n (F_i^{-l}(x_{(l)}) - y_{(l)}) (F_j^{-l}(x_{(l)}) - y_{(l)}). \quad (37)$$

In the averaging based on the Stacking approach (Wolpert 1992, Breiman 1996, Cherkassky & Mulier 2007) the weights are selected by solving a simple optimization problem:

$$w = \arg \min_w \sum_{j=1}^n \left( y_{(j)} - \sum_{i=1}^b (w_i F_i^{-j}(x_{(j)})) \right)^2. \quad (38)$$

This optimization problem can be solved using the OLS. However, it is expected that the predictive accuracy of the ensemble will increase when also (34) and  $w_i \geq 0$ ,  $i=1,2,\dots,b$  are satisfied (Breiman 1996, LeBlanc & Tibshirani 1993). To solve (38) with the non-negativity constraints, the NNLS is used (note that VariReg solves the NNLS problem using Matlab requiring it to be installed on the computer).

It is worth noting that the models used for combination are recommended to be diverse. In VariReg any of the individual regression modelling methods may be used for combination – full and sparse polynomials, ABFC, LWP, k-NN, RBF, Kriging, MARS, and PNN.

### 3. IMPLEMENTED OPTIMIZATION ALGORITHMS

VariReg provides means for single-objective optimization (either minimization or maximization) of the values of output variables using the built regression models as objective functions. In the current version of VariReg, the following optimization algorithms are implemented: Particle Swarm Optimization (PSO) and a simple Grid Search (GS).

#### 3.1. Particle Swarm Optimization

PSO was initially proposed by Kennedy and Eberhart (Kennedy & Eberhart 1995). It is a stochastic population-based global optimization algorithm inspired by the social dynamics of flocking organisms, which employs a swarm of particles to probe the search space.

A particle of the swarm is completely described by its current position in the search space, its velocity vector, a memory of its best previous position, and information regarding the best position ever attained by the swarm. Using this information, the velocity vector is used to update the position of each particle in the swarm (Kennedy & Eberhart 1995, Michalewicz 1995, Marler & Arora 2004, Rojas et al. 2004).

The initial swarm is created by distributing the particles randomly throughout the search space. The initial position the initial velocity vectors of particles indexed by  $i$  are given by the following equations:

$$x_i(0) = x_{\min} + r_1(x_{\max} - x_{\min}) \quad (39)$$

$$v_i(0) = x_{\min} + r_2(x_{\max} - x_{\min}) \quad (40)$$

where  $r_1$  and  $r_2$  are random vectors with components uniformly distributed in the interval  $[0,1]$ ;  $x_{\min}$  is the lower bound vector and  $x_{\max}$  is the upper bound vector for the variables.

At each optimization step, each particle adjusts its position and velocity according to the following equations:

$$x_i(t+1) = x_i(t) + v_i(t+1) \quad (41)$$

$$v_i(t+1) = wv_i(t) + c_1r_1(p_i - x_i(t)) + c_2r_2(p_s - x_i(t)) \quad (42)$$

where  $t$  indicates iterations;  $r_1$  and  $r_2$  are random vectors with components uniformly distributed in the interval  $[0,1]$ ;  $p_i$  is the best position found by the  $i$ th particle so far and  $p_s$  is the best position found by any particle so far; the parameters  $c_1$  and  $c_2$  are called cognitive and social parameters, respectively, and they are used to bias the particle's search towards its own best previous position and towards the best previous position of the swarm; the parameter  $w$  is called inertia weight and is used to control the trade-off between the exploration and the exploitation ability of the swarm. Note that in the case when the velocity of a particle would move it outside the search space, the particle is moved back up to the border of the space. Also the velocities of the particles are not allowed to be higher than  $(x_{\max} - x_{\min})$ .

Regarding parameters  $c_1$  and  $c_2$  the literature (Kennedy & Eberhart 1995, Michalewicz 1995, Marler & Arora 2004, Rojas et al. 2004) proposes using either  $c_1 = c_2 = 2$  or other values satisfying  $c_1 + c_2 = 4$ . For inertia weight  $w$ , generally the value is proposed somewhere around 1. However, the value of  $w$  may also be dynamically reduced, e.g. from 1 to 0.35. In VariReg the value of  $w$  is at first initialized with  $w_1$  and then in each optimization step the value is reduced by  $\Delta w$  until in the very last step  $t_{\max}$  it reaches  $w_2$  ( $w_2 \leq w_1$ ), i.e. the updating scheme for  $w$  is

$$w(t+1) = w(t) - \Delta w \quad (43)$$

where  $w(0) = w_1$  and  $\Delta w = (w_1 - w_2)/t_{\max}$ . While the maximum number of performed iterations  $t_{\max}$  is defined by the user, the algorithm also stops when all the particles are at the same position.

Additionally, to reduce the probability of not obtaining the globally optimal solution due to convergence to a local optimum, VariReg offers an additional parameter – “number of restarts” which determines how many times the whole PSO algorithm must be completely restarted to obtain a possibly better solution.

### **3.2. A simple Grid Search**

GS is a trivial algorithm intended for use in very low dimensionalities. The  $y$  values are probed in a uniform grid fineness of which is defined by the number of the considered values for each dimension. The advantage of GS is that it is completely deterministic meaning that given the number of values for the variables it will always produce the same result.

## 4. VARIREG USER'S INTERFACE

### 4.1. Input file format

See Figure 2 and Figure 3 for explanations of VariReg's input file format (for training data and test data correspondingly). Note that after loading of training data set, it is possible also to take some percentage of data points from the training data set as Hold-Out data for testing or use the data with Cross-Validation.

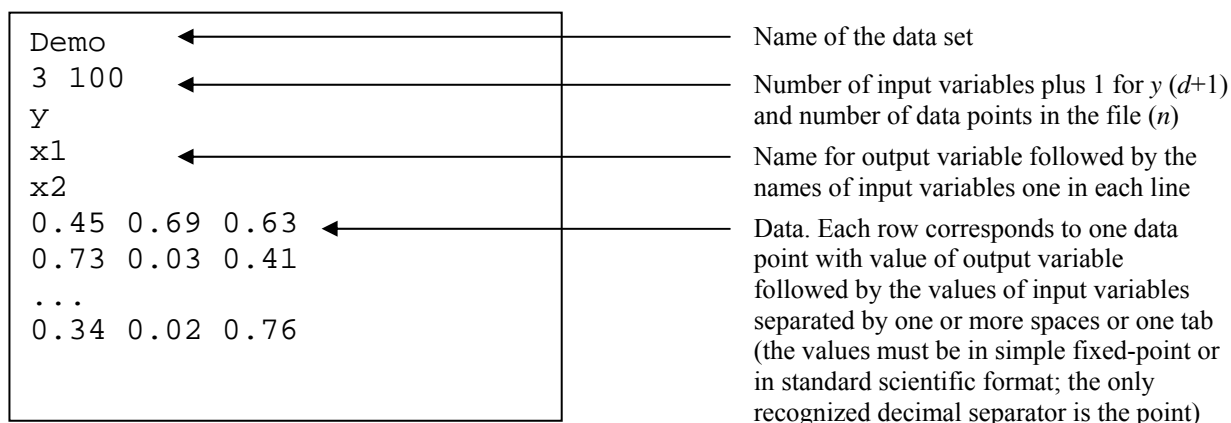


Figure 2. Input file format – the training data

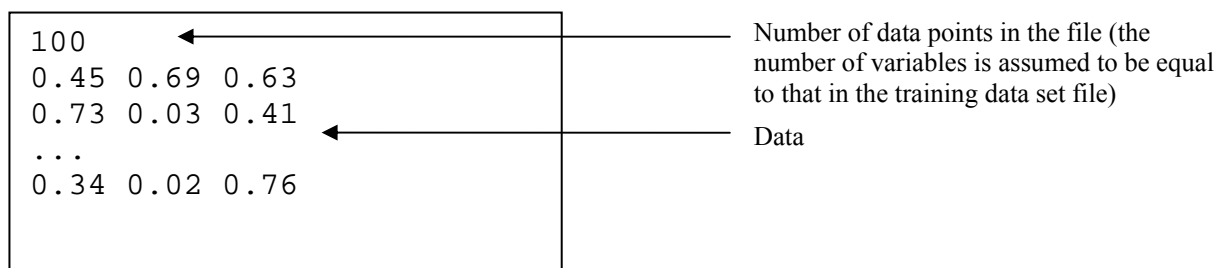


Figure 3. Input file format – the test data

### 4.2. User's interface

VariReg user's interface consists of a window with tabs and a log. There are the following tabs:

- tab "Data" – manipulation of training and test data sets as well as Cross-Validation for evaluation of the modelling methods;
- tab "Polynomials" – full and sparse polynomials built by subset selection methods;
- tab "ABFC" – modelling using Adaptive Basis Function Construction;
- tab "LWP" – modelling using Locally Weighted Polynomials;
- tab "k-NN" – modelling using k-Nearest Neighbours;
- tab "RBF" – modelling using Radial Basis Function interpolation;
- tab "Kriging" – modelling using Kriging interpolation;
- tab "MARS" – modelling using MARS;
- tab "PNN" – modelling using Polynomial Neural Networks induced by Group Method of Data Handling;
- tab "Averaging" – modelling using model averaging/ensembling/combining;
- tab "Optimization" – optimization of the values of output variables using the built regression models as objective functions;

- tab “Information” – information about VariReg’s version, copyright, author’s contact e-mail and webpage address, as well as citations for references.

#### 4.2.1. Tab “Data”

Figure 4 demonstrates user’s interface of the tab:

1. Button for loading training data set.
2. For all types of Cross-Validation it is important that the order of the data points in the whole training data set is uninformative (randomized). This checkbox provides the possibility to shuffle the order of the data points in the training data set right after loading it. The “random seed” enables the shuffling to be identical if multiple modelling methods are tested and the training data is reloaded.
3. Button for saving the (shuffled) training data set.
4. Radio button for selection of Hold-Out type of testing of modelling methods – in this case the user must load the test data set from additional file or take some percentage of data points from the training data set.
5. Radio button for selection of  $\nu$ -fold Cross-Validation type of testing of modelling methods – in this case the user must set the number of folds  $\nu$  for the Cross-Validation and each time when any modelling algorithm is started, the program will perform Cross-Validation (i.e. the modelling will be restarted  $\nu$  times) and the results will be stored in a user-chosen file which can be used for further calculations of averages, variances, standard deviations etc. Note that the Cross-Validation is done in the proper way – cross-validated are the modelling methods themselves not the models (i.e. the Cross-Validation loop is performed over the whole modelling method). The modelling process will be done  $\nu$  times. Note also that the models built during the Cross-Validation should not be used any further – to build models for further applications the “Hold-out test data set (or no testing)” radio button should be checked.
6. Radio button for selection of Hold-Out type of testing of modelling methods using a series of train/test files – in this case the user must supply template filenames (together with file paths) for the train and test files. When the modelling is started, a string “#.txt” (where # is the number of the file) will be added to these names automatically and the corresponding files will be loaded. This means that if the user has data files e.g. “train1.txt”, “train2.txt”, “train3.txt”, “test1.txt”, “test2.txt” and “test3.txt” on drive “C:”, then the templates must look like “C:\train” and “C:\test” and the values in fields “From” and “To” must be 1 and 3 correspondingly. In case the “Use one test file for all training files” is checked, for all the training data files a single one test data file will be used (and no additional symbols will be added to the file path, e.g. the entered string should look like “C:\test.txt”).
7. A small log for information on training data and testing data – file names, number of data points (data cases), number of input variables, mean value of  $y$ , variance of  $y$ , standard deviation of  $y$  etc.

Note that in Cross-Validation or “series of train/test files” modes, VariReg will be creating a number of temporary files in either its own directory or in the directory of data set. Therefore, for speed considerations, it is not recommended to use VariReg from a slow storage device like flash memory etc.

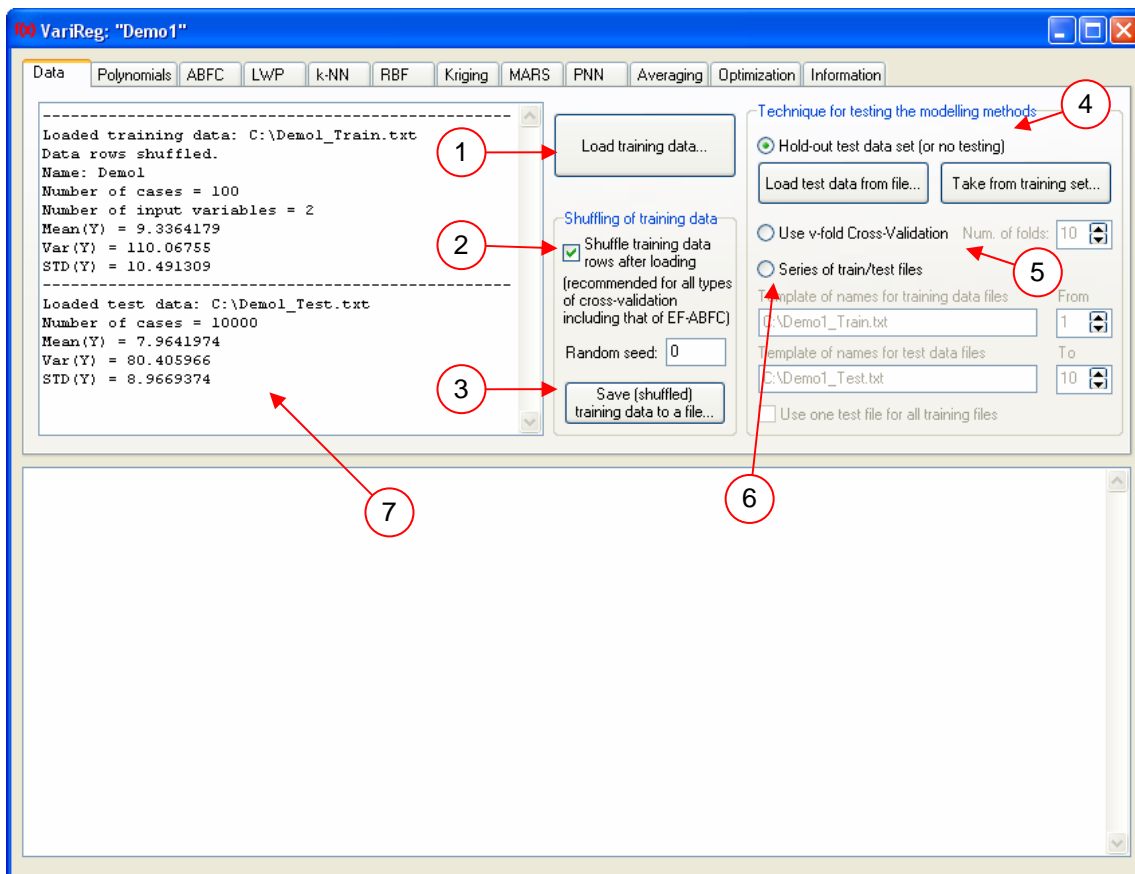


Figure 4. Tab “Data”

#### 4.2.2. Tab “Polynomials”

Figure 5 demonstrates user’s interface of the tab:

1. (Maximal) degree of the model (in the box “Funcs” the total number of the basis function, considering the number of input variables and the degree, is shown).
2. Radio buttons for selection of full polynomials (no subset selection) or one of the subset selection algorithms.
3. Radio buttons for selection of criterion for subset selection.
4. The list of built models – user can click on any of the models to get information on it as well as to draw its 3D surface or to output its functional form.
5. Whether or not the subset selection will start with the intercept term already included (and fixed) in the model.
6. Button for starting the search for the best model.
7. Button for drawing a 3D surface of the selected model (see Figure 6).
8. Button for clearing the log.
9. Whether the degree of the models must be found automatically via Cross-Validation (requires considerably larger computational resources).
10. Button for saving predictions (in the training and/or test data sets) of the model to a file.
11. Button for outputting the equation of the model in such way that it can be copied into MS Excel or other spreadsheet software for further calculations.
12. Information on the just built or just selected model:
  - 1) “Total number of basis functions” – the total number of basis functions which were allowed to be used for model building;
  - 2) “Used basis functions” – the number of basis functions actually included in the model;
  - 3) “Total number of input variables” – the total number of input variables in the data set;

- 4) “Number of used input variables” – the number of input variables actually used in the model;
- 5) “Degree” – degree of the model (the largest degree of all the basis functions);
- 6) “TrainMSE” – MSE error in the training data set;
- 7) “TestMSE”, “TestRRMSE”, “TestR2” – MSE, RRMSE, and  $R^2$  errors in the test data set;
- 8) “Used criterion” – criterion used in model building;
- 9) “Crit value” – value of the used criterion for the model;
- 10) “Iterations” – the number of the performed iterations of the chosen search algorithm;
- 11) “Model evaluations” – the number of model evaluations performed by the chosen search algorithm;
- 12) “Method” – name of the chosen modelling method or search algorithm;
- 13) “Time (s)” – time consumption of the chosen algorithm in seconds (not including data loading time, model testing time and any other unrelated processing);
- 14) “Func=” – each basis function of the model is showed as a sequence of degrees for each input variable of the data set in the same order as the variables appear in the input files;
- 15) “Coef=” – coefficient value for a used basis function (if a basis function is not used in the model, it has no coefficient showed).

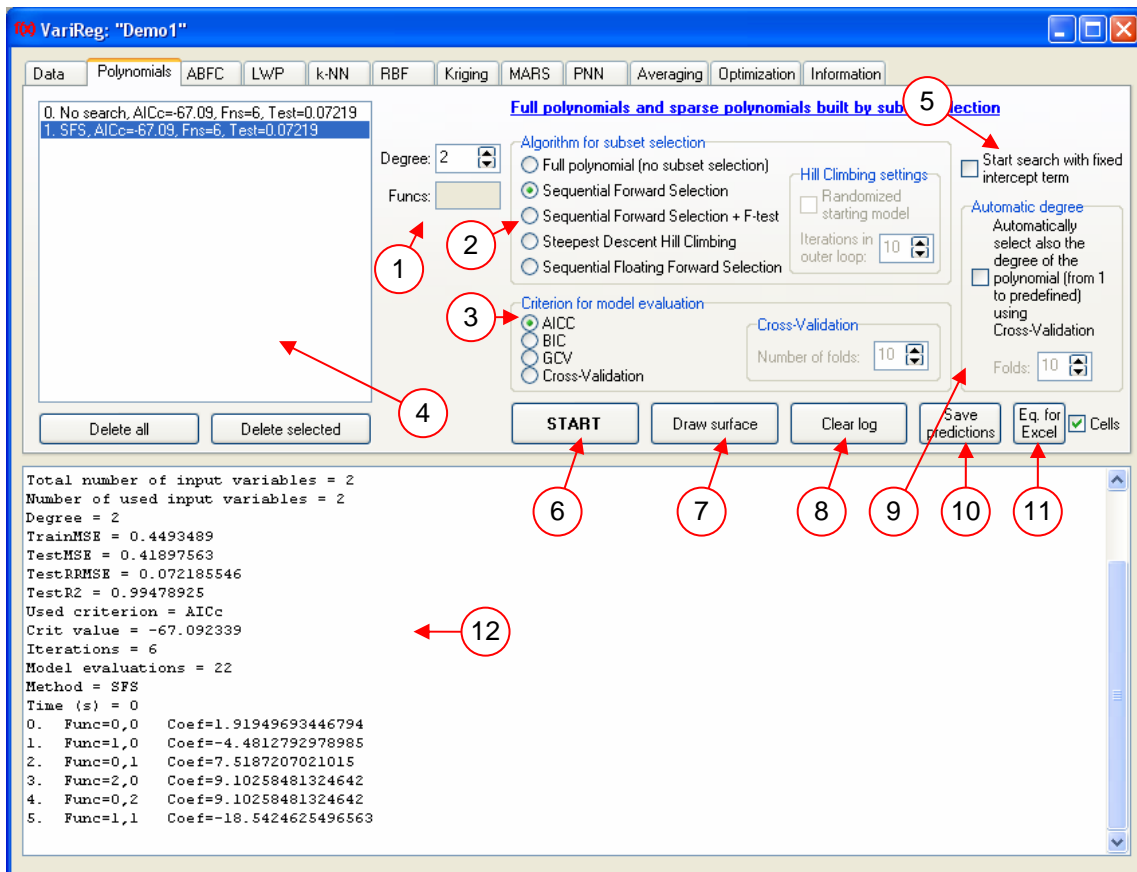


Figure 5. Tab “Polynomials”

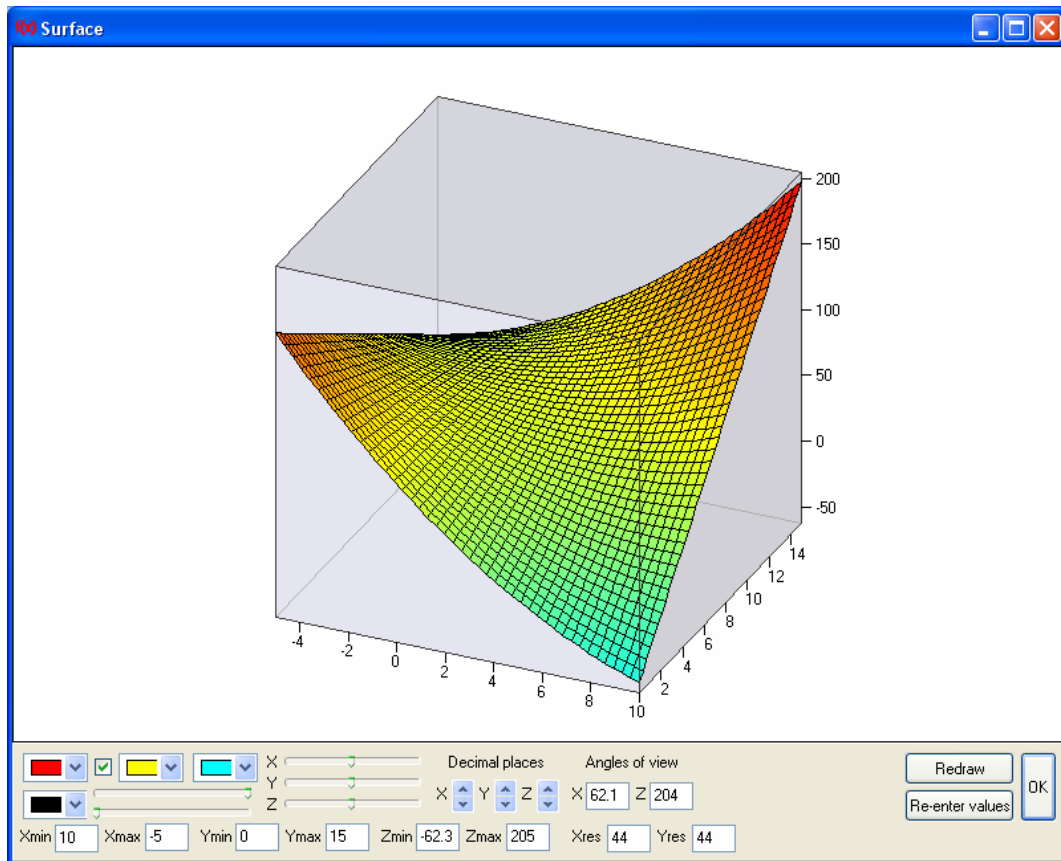


Figure 6. Window displaying 3D surface of a selected model

#### 4.2.3. Tab “ABFC”

Figure 7 demonstrates user’s interface of the tab:

1. Whether to perform EF-ABFC over F-ABFC (the process will take longer, however, the obtained model mostly will be more reliable).
2. Optional settings for F-ABFC:
  - 1) whether to use the refinement operator which increases individual exponents (this is from an older F-ABFC version and mostly is not required);
  - 2) whether to use higher recursion depth for the refinement operators (this is intended mostly for regression problems with low number of input variables as in these situations the search algorithm of F-ABFC may sometimes get stuck in a too early local minima);
  - 3) whether to limit model’s degree (mostly not required).
3. Radio buttons for selection of criterion for subset selection.
4. Information on the just built or just selected model (see Section 4.2.2. Point 11).

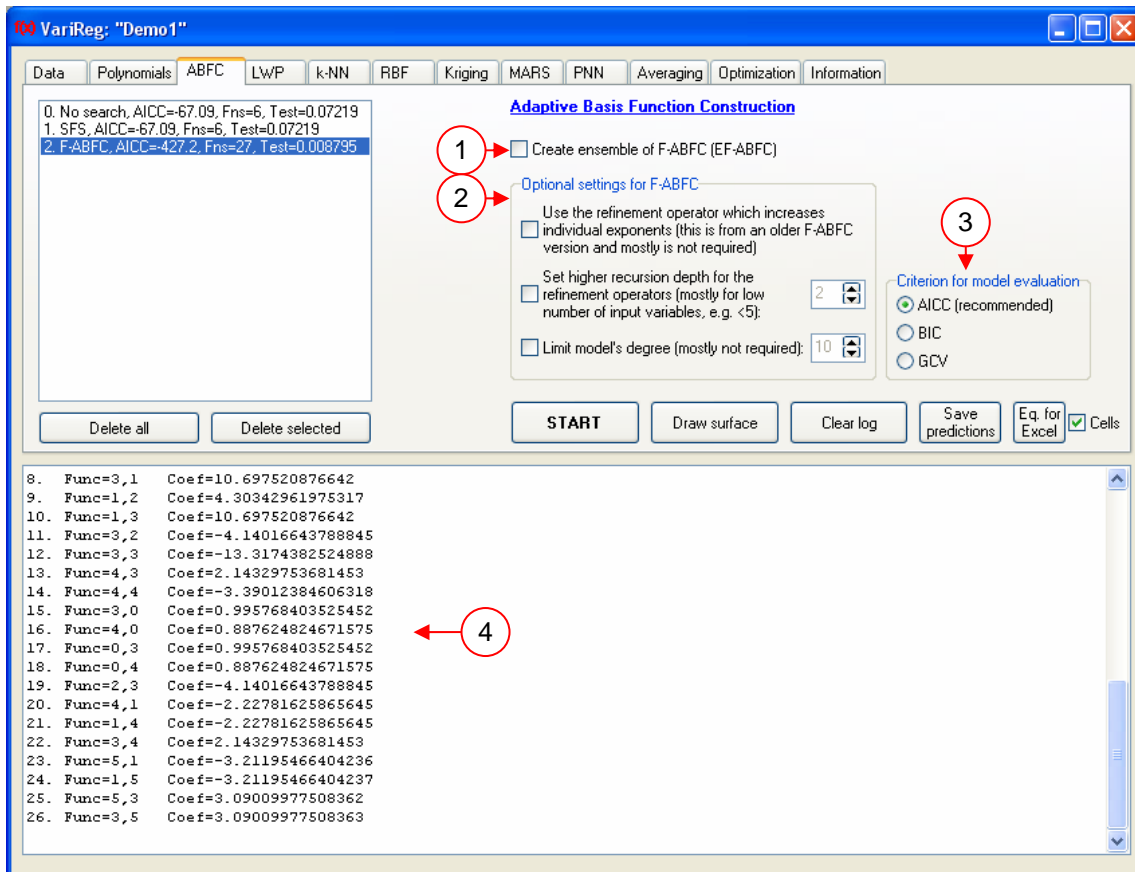


Figure 7. Tab “ABFC”

#### 4.2.4. Tab “LWP”

Figure 8 demonstrates user’s interface of the tab:

1. Radio buttons for selection of one of the two types of LWP described in Section 2.5.
2. Radio buttons for manual or automatic (using LOOCV) bandwidth selection.
3. Whether or not the degree also must be selected automatically using LOOCV.
4. Radio buttons for selection of scaling of the inputs for all the data points (in training as well as and in testing data). The scaling is available only when the distance to the farthest point is not taken into account as otherwise it has no effect on the results. Also note that the data will be scaled only for the moment of modelling and testing.
5. Information on the just built or just selected model:
  - 1) “Total number of basis functions” – the number of the basis functions in the model (for LWP it is equal to the total number of the basis functions);
  - 2) “Total number of input variables” – the total number of input variables (equal to the number of input variables included in the model);
  - 3) “Bandwidth” – chosen bandwidth of the LWP;
  - 4) For other see Section 4.2.2. Point 11.

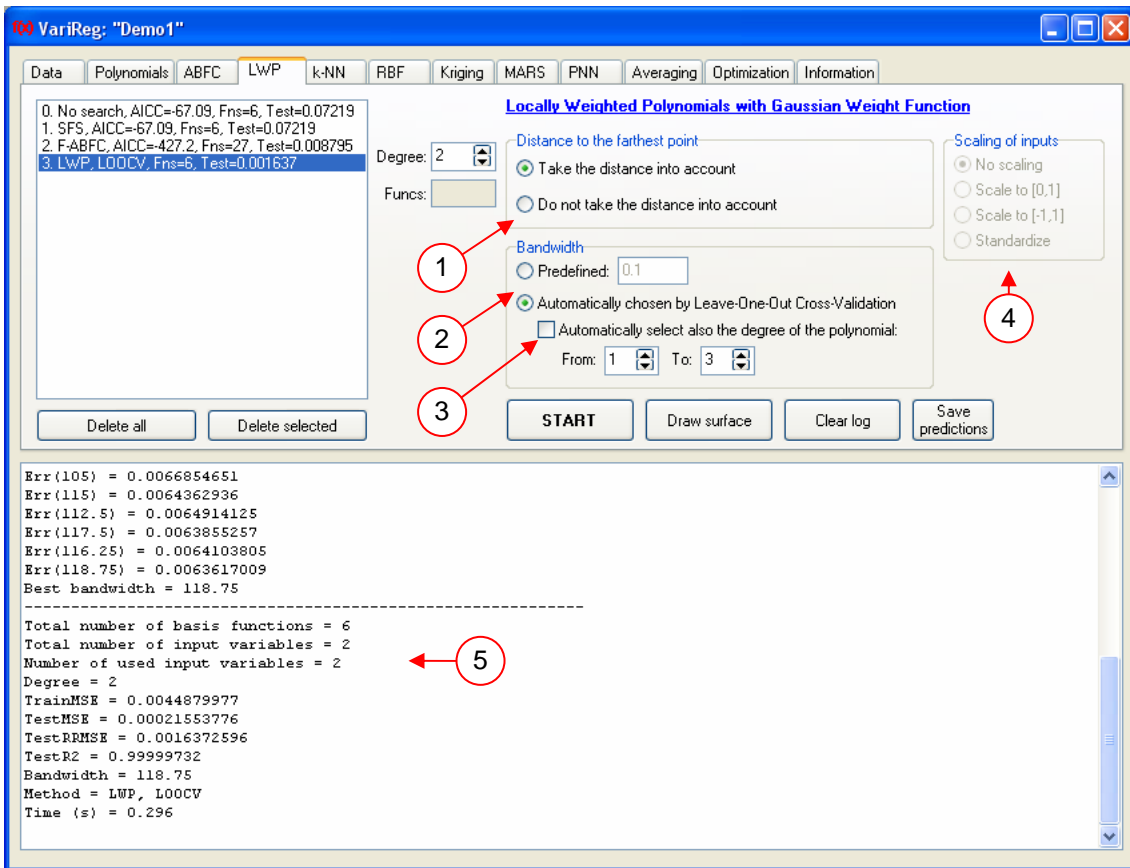


Figure 8. Tab “LWP”

#### 4.2.5. Tab “k-NN”

Figure 9 demonstrates user’s interface of the tab:

1. The number of the nearest neighbours used for prediction.
2. Radio buttons for selection of the distance weighting scheme.
3. Radio buttons for selection of scaling of the inputs for all the data points (in training as well as and in testing data). Note that the data will be scaled only for the moment of modelling and testing.
4. Information on the just built or just selected model (see Section 4.2.2. Point 11).

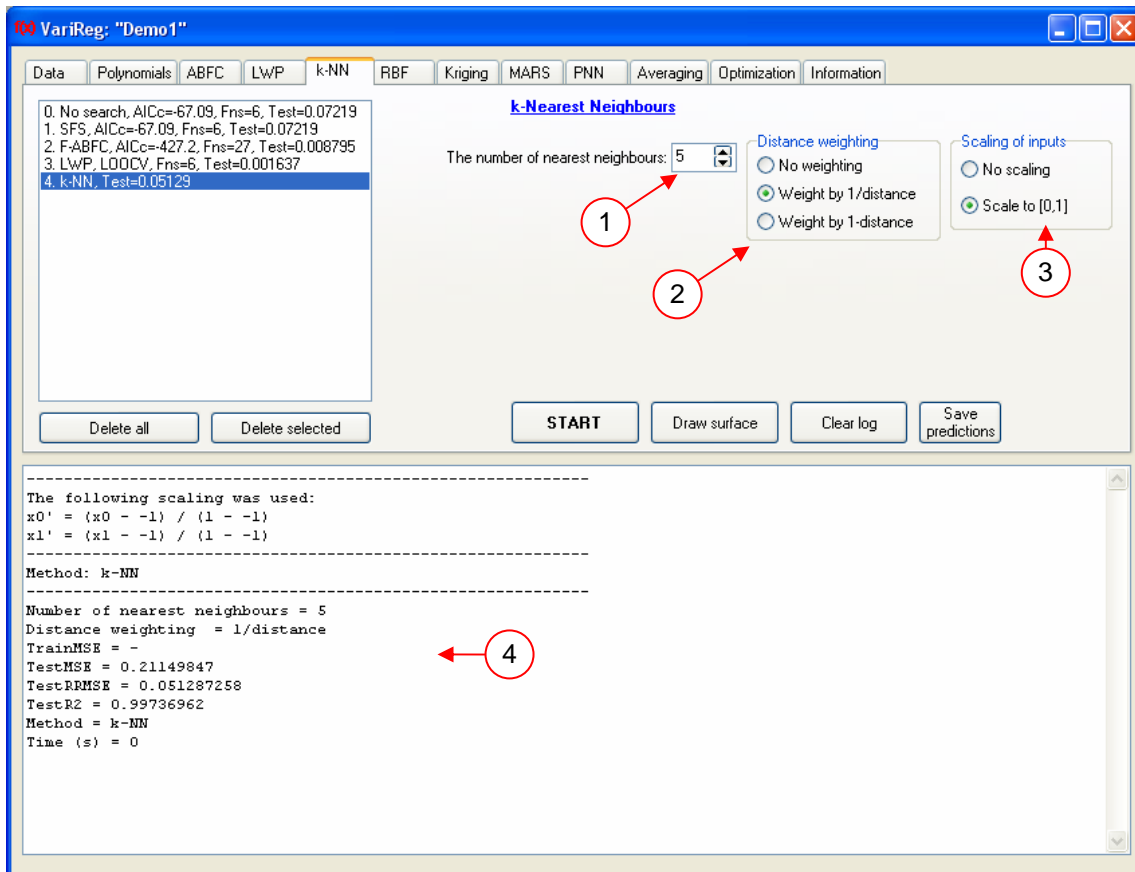


Figure 9. Tab “k-NN”

#### 4.2.6. Tab “RBF”

Figure 10 demonstrates user’s interface of the tab:

1. Radio buttons for selection of the type of basis functions.
2. Radio buttons for manual or automatic (using LOOCV) smoothing parameter selection for Gaussian RBF.
3. Radio buttons for selection of parameter  $c$  for multiquadric and thin plate spline RBF.
4. Radio buttons for selection of scaling of the inputs for all the data points (in training as well as and in testing data). Note that the data will be scaled only for the moment of modelling and testing. Also note that the model parameters displayed will be those calculated according to the scaled data.
5. Information on the just built or just selected model:
  - 1) “Total number of basis functions” – the number of basis functions in the model (equal to the number of data points in the training data set);
  - 2) “RBF type” – RBF type;
  - 3) “C” – value of the shape parameter for multiquadric or thin plate spline basis functions;
  - 4) “RBF smoothing parameter” – value of the Gaussian RBF smoothing parameter;
  - 5) “Power-law function parameter” – value of the parameter in the power-law function;
  - 6) “Coef=” – coefficient value for a basis function (each coefficient and basis function corresponds to one point in the training data, ordered in the same way the training data is ordered – note that the order may be shuffled if the “shuffle” checkbox was checked when the data was loaded);
  - 7) For other see Section 4.2.2. Point 11.

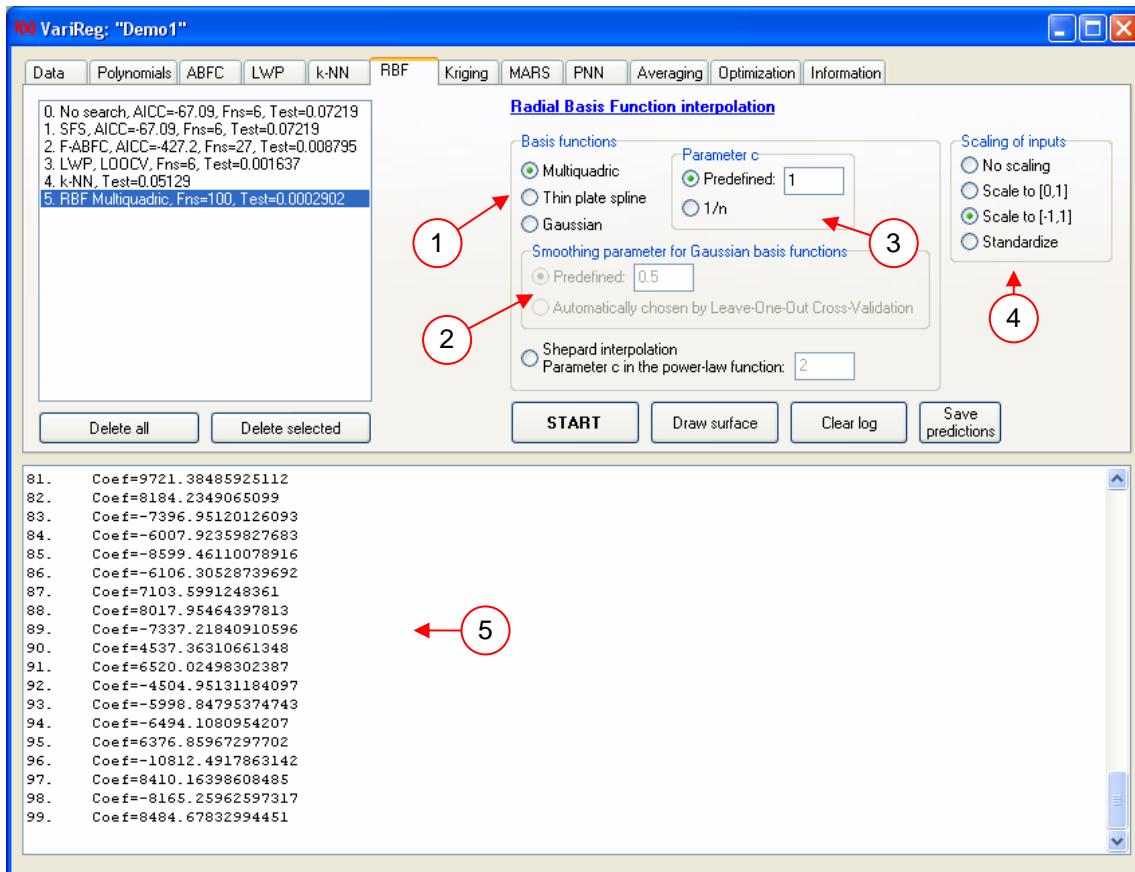


Figure 10. Tab “RBF”

#### 4.2.7. Tab “Kriging”

Figure 11 demonstrates user’s interface of the tab:

1. Radio buttons for selection of the degree of polynomial in the Kriging model.
2. Radio buttons for selection of the correlation function type in the Kriging model.
3. The user must provide initial guess, as well as lower bound and upper bound of correlation function parameters.
4. The user must provide a path to the `matlab.exe` (required for the Kriging method implementation to work). The user must only browse for the `matlab.exe` file – no additional configuration of Matlab or special installation of the toolbox is required.
5. Draw surface of the model right after building it.
6. Checkbox for saving the predictions (in the training and/or test data sets) of the built model to a file right after building it.
7. Information on the just built model (see Section 4.2.2. Point 11).

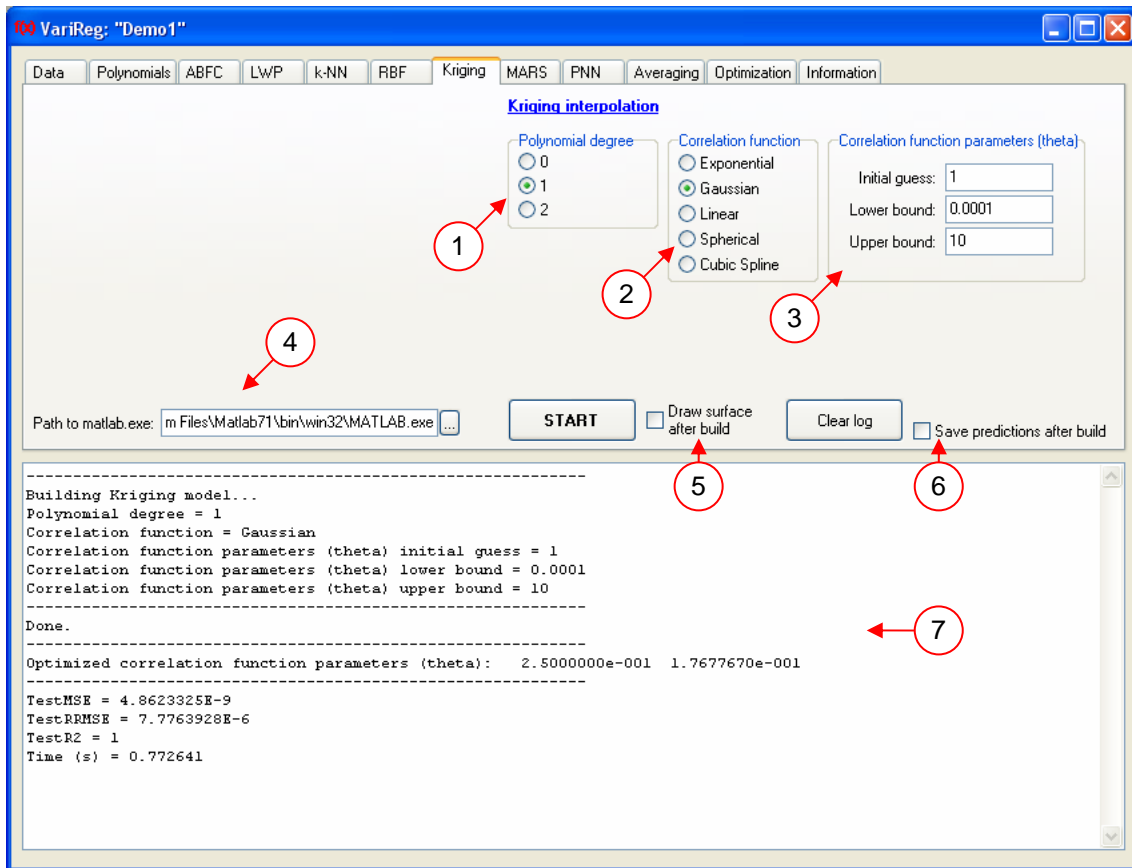


Figure 11. Tab “Kriging”

#### 4.2.8. Tab “MARS”

Figure 12 demonstrates user’s interface of the tab:

1. Options for the MARS method (see MARS overview in Section 2.9 for details).
2. Choosing the best value for the number of degrees of freedom charged for knot optimization using 10-fold Cross-Validation.
3. Draw surface of the model right after building it.
4. Checkboxes for seeking optimum of the output variable using the built model and saving the predictions (in the training and/or test data sets) of the built model to a file right after building it.
5. Information on the just built model (see Section 4.2.2. Point 11).

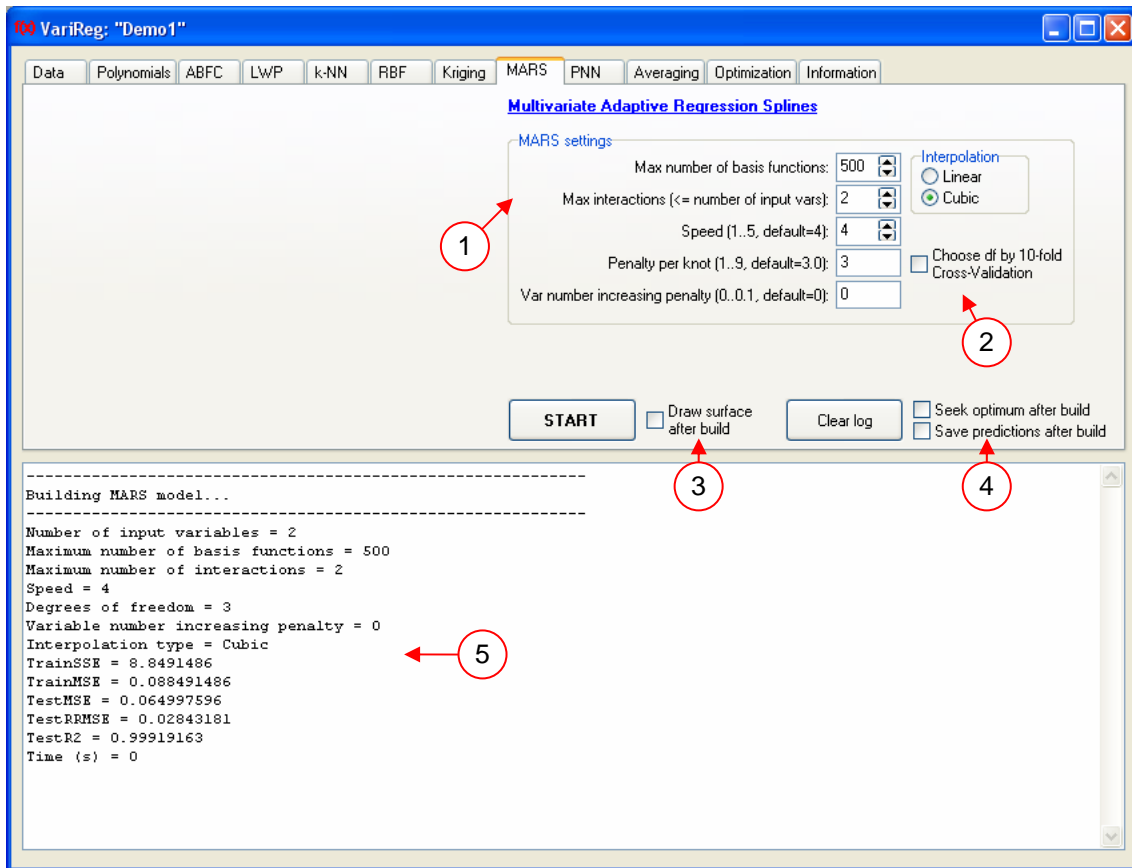


Figure 12. Tab “MARS”

#### 4.2.9. Tab “PNN”

Figure 13 demonstrates user’s interface of the tab:

1. (Maximal) degree of polynomials in each neuron.
2. Radio buttons for selection of full polynomials (no subset selection) or one of the subset selection algorithms for generating neurons.
3. Radio buttons for selection of criterion for subset selection in each neuron as well as for deciding when to stop the building of the network.
4. Radio buttons for selection of whether the inputs to the neurons are taken only from the immediately preceding layer or also from the original input variables.
5. Maximum number of inputs for each neuron.
6. Maximum number of neurons in each layer.
7. Draw surface of the model right after building it.
8. Checkboxes for seeking optimum of the output variable using the built model and saving the predictions (in the training and/or test data sets) of the built model to a file right after building it.
9. Information on the just built model:
  - 1) “Total number of generated layers” – the total number of the generated layers of the network (the last layer is discarded);
  - 2) “Number of layers” – the number of layers in the final network;
  - 3) “Used input variables” – the list of input variables used in the final network;
  - 4) “The number of used input variables” – the number of input variables used in the final network;
  - 5) “Crit value” – used criterion’s value for the final network;
  - 6) For other see Section 4.2.2. Point 11.

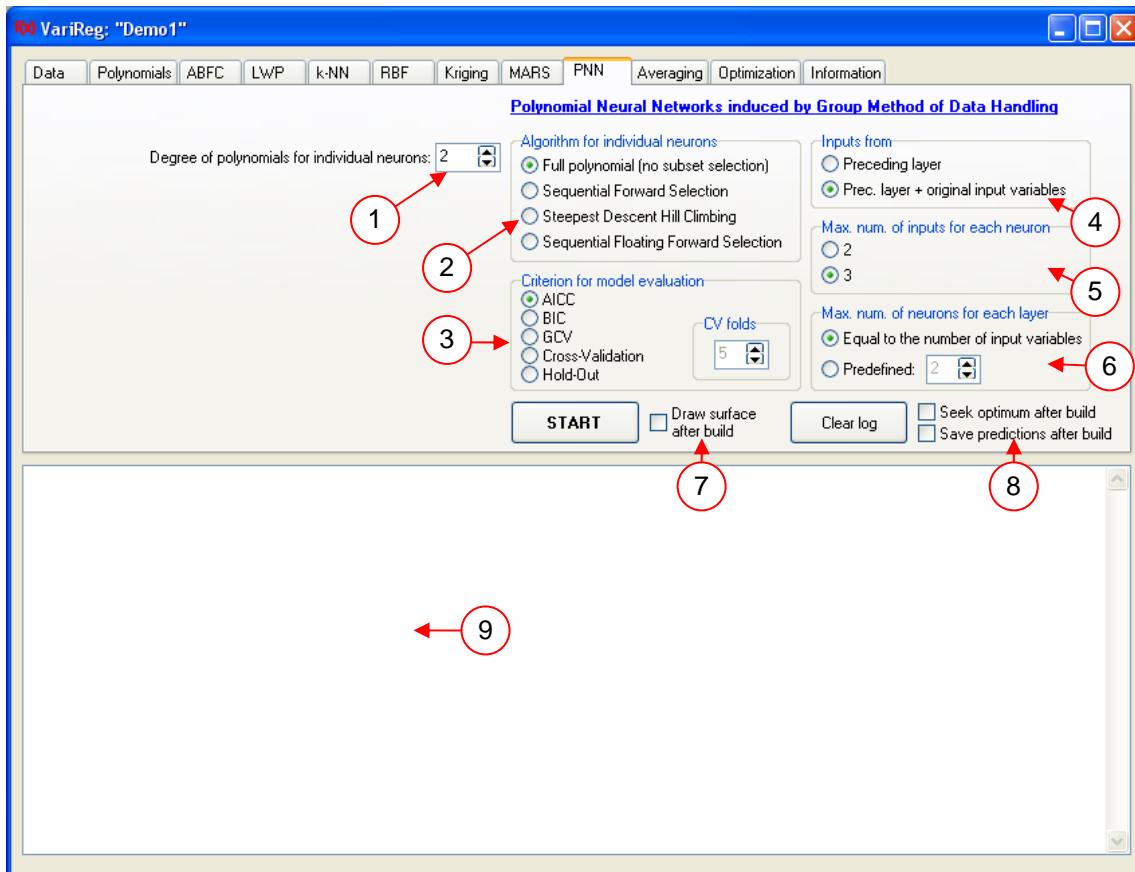


Figure 13. Tab “PNN”

#### 4.2.10. Tab “Averaging”

Figure 14 demonstrates user’s interface of the tab:

1. Selection of the modelling methods used for averaging.
2. Selection of the averaging methods.
3. The user must provide a path to the `matlab.exe` (required for the Kriging method and the NNLS method implementations to work). The user must only browse for the `matlab.exe` file – no additional configuration of Matlab or special installation of the toolbox is required.
4. Draw surface of the model combination from a one averaging method right after building the combination.
5. Checkbox for saving the predictions (in the training and/or test data sets) of the model ensemble to a file right after building it.
6. Information on the just built model combinations:
  - 1) LOOCV errors and error variances;
  - 2) predictive performances for the individual methods in the test data set;
  - 3) models weights calculated by the averaging methods;
  - 4) predictive performances for the averaging methods in the test data set;
  - 5) total elapsed time.

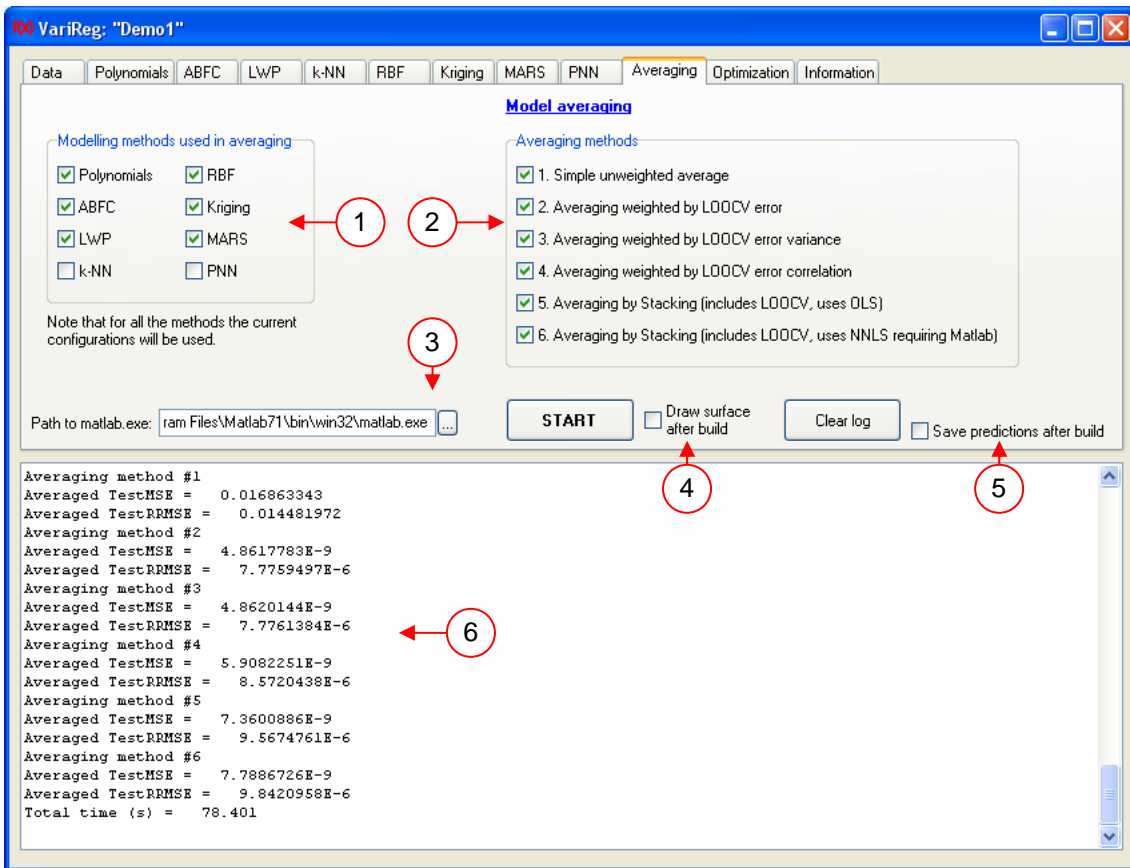


Figure 14. Tab “Averaging”

#### 4.2.11. Tab “Optimization”

Figure 15 demonstrates user’s interface of the tab:

1. The list of built models – user can click on any of the models to get information on it, to draw its 3D surface as well as to perform optimization using this model.
2. Choice of the optimization problem type – minimization or maximization.
3. Choice of the optimization algorithm.
4. Particle Swarm Optimization algorithm settings.
5. Grid Search algorithm settings.
6. Button for starting the optimization.
7. Optimization results (values of input variables for the optimized value of output variable as well as the number of performed function evaluations and elapsed time).

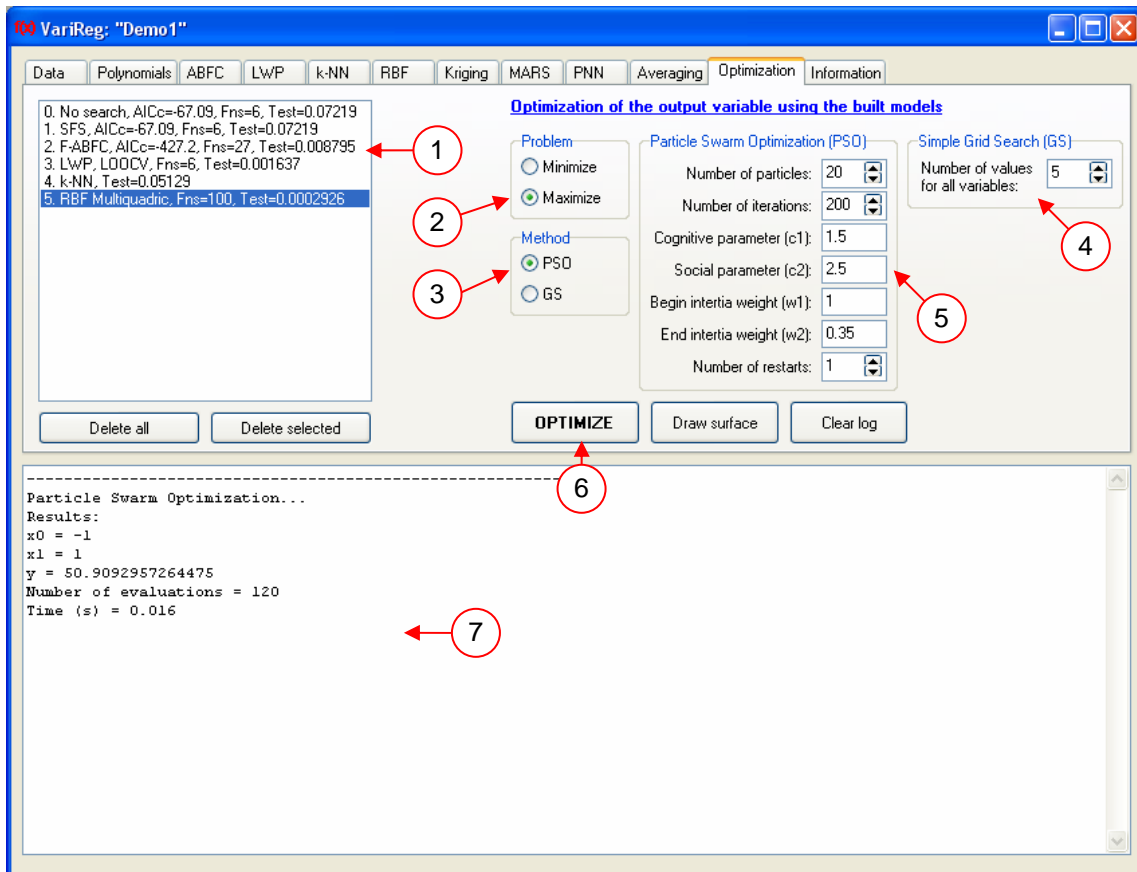


Figure 15. Tab "Optimization"

## 5. USING VARIREG FROM COMMAND LINE AND FROM MATLAB

All the regression modelling methods implemented in VariReg can also be accessed from:

- command line using `.ini` configuration files;
- Matlab environment using wrapper functions provided together with VariReg.

### 5.1. Using VariReg from command line

To use VariReg methods from a command line you must do the following:

- Make an `.ini` file which contains all the necessary information for VariReg to self-configure its parameters.

File `ParamsFullDefault.ini` provided together with VariReg contains all the available VariReg parameters with their names and default values. In this file every parameter is explained. All the parameters correspond to visual controls in the VariReg user's interface so also all the explanations of how to configure VariReg's methods given in sections 3 and 4 of this user's manual apply also to the parameters in the `.ini` files.

- Prepare your training and/or test data files in the right format.

The format of training and test data files is similar to VariReg's "normal" input files (explained in figures 2 and 3) except that the files must not contain headers, i.e. a data file starts immediately with the input-output values of the first data point (see Figure 16).

0.45	0.69	0.63
0.73	0.03	0.41
...		
0.34	0.02	0.76

Figure 16. Input file format for both training data and test data; each row is one data point

If testing the model using a test data set is not required but instead it has to be applied for prediction of unknown values, the column of output variable  $y$  in the test file then may contain just zeros or any other random values as this column does not affect the prediction process (it is only required for correct calculation of error measures which are then stored in one of the output files).

- Execute `VariReg.exe` with a command line parameter which informs the program about the location of your `.ini` file (for example it can be done using a `.bat` file or from another program).

If for example `VariReg.exe` is located at `C:\VariReg\` and the configuration parameters file `MyParams.ini` is located at `C:\` then the command line will look like this: `C:\VariReg\VariReg.exe "C:\MyParams.ini"`. Note that the complete path to the `.ini` file is recommended to be put in quotation-marks as otherwise spaces in file or directory names will disrupt the path.

- Read-in the results from the VariReg output files.

VariReg saves the chosen modelling method's predictions in two separate files – one for predictions in training data and one for predictions in test data. The format of the output files is similar to the format of the input files except that the first columns correspond to input variables  $x$ , the second last column corresponds to output variable  $y$  (values of which were given in the input file), and the last column contains values which were predicted by the built model (see Figure 17 compared to Figure 16). Note that you will get output files formatted in the same way if you use the "Save predictions" button (or checkbox) when the program is used in direct manner from the user's interface.

0.69	0.63	0.45	0.44
0.03	0.41	0.73	0.74
...			
0.02	0.76	0.34	0.32

Figure 17. Output file format for both training data and test data; each row corresponds to one data point with its actual and predicted  $y$  value (compare to input file format given in Figure 16)

Additionally VariReg can create also an “information” file which has just the same content as the one displayed in the VariReg user’s interface log when the modelling method is executed (e.g., when the program is used in direct manner). The information file can be further used to e.g. extract the actual execution time of the modelling method or extract its accuracy evaluation results.

If using `.ini` file the VariReg was configured to perform an outer Cross-Validation loop (to evaluate the modelling method’s predictive performance), only an “information” file will be outputted containing all the evaluation information gathered in each Cross-Validation iteration. Here the format of such an output file is dependent of the used modelling method. The specific format can be found out by doing Cross-Validation directly from the VariReg user’s interface on any training data file using the same method and saving the results file.

- Delete the output files.

See `.ini` and `.bat` file examples provided together with VariReg.

## 5.2. Using VariReg from Matlab

To use VariReg methods from Matlab you must do the following:

- Add the directory which contains the wrapper functions (“`...\VariReg\Using VariReg` from command line and from `Matlab\Matlab\`”) to the search path of Matlab.
- Edit the file `variireg_getpath.m` so that it contains the actual path to `VariReg.exe`. By default it is assumed that the directory is “`C:\VariReg\`”. All the provided wrapper functions will place also all their temporary input and output files in this directory.
- Use the wrapper functions in `.m` files provided together with VariReg or create new functions for specific needs. In general, using VariReg’s methods from Matlab’s wrapper functions is very similar to using the methods from command line, i.e. 1) a configuration file for VariReg is created; 2) the training/test data is saved in temporary files; 3) VariReg is run; 4) results are read-in; 5) temporary files are deleted. The contents of all the files are formatted in the same way as explained in Section 5.1. However, those details can be just ignored when you are calling the wrapper functions themselves.

## 6. REFERENCES

1. Acar E., Rais-Rohani M. Ensemble of metamodels with optimized weight factors, *Structural and Multidisciplinary Optimization*, Vol. 37, No. 3, Springer, 2009, pp. 279-294
2. Anastasakis L., Mort N. The development of self-organization techniques in modelling: a review of the Group Method of Data Handling (GMDH), Tech. report 813, University of Sheffield, Department of Automatic Control & Systems Engineering, Sheffield, UK, 2001
3. Barron R.L., Mucciardi A.N., Cook F.J., Craig J.N., Barron A.R. Adaptive Learning Networks: Development and Application in the United States of Algorithms Related to GMDH, Ch. 2. Self-Organizing Methods in Modeling: GMDH Type Algorithms. Farlow S.J. (ed.), Marcel Dekker, NY, 1984, pp. 25-65
4. Bishop C.M. Neural networks for pattern recognition. Oxford University Press, New York, 1995, pp. 364-369.
5. Breiman L. Stacked Regressions, *Machine Learning*, Vol. 24, 1996, pp. 49-64
6. Cherkassky V., Mulier F.M. Learning from Data: Concepts, Theory, and Methods, 2nd ed. Wiley-IEEE Press, 2007, 538 p.
7. Cleveland W. S., Devlin S. J. Locally weighted regression: An approach to regression analysis by local fitting, *Journal of the American Statistical Association*, Vol. 83, 1988, pp. 596-610
8. Colaco M.J., Dulikravich G.S. A Hybrid RBF Based Method for Highly Multidimensional Response Surfaces Using Scarce Data Sets, 12th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Victoria, British Columbia Canada, 2008
9. Colaco M.J., Dulikravich G.S., Sahoo D. A comparison of two methods for fitting high dimensional response surfaces, *Inverse Problems, Design and Optimization Symposium*, Miami, Florida, USA, 2007, 8 p.
10. Craven P., Wahba G. Smoothing noisy data with spline functions. Estimating the correct degree of smoothing by the method of generalized cross-validation, *Numerische Mathematik*, Vol 31, 1979, pp. 317-403
11. Elder IV J.F., Brown D.E. Induction and Polynomial Networks. *Network Models for Control and Processing*, Fraser M.D. (ed.), Portland, OR, Intellect, 2000, pp. 143-198
12. Farlow S.J. Self-organizing Methods in Modeling: GMDH Type Algorithms, 54. Marcel Dekker, Inc., NY, 1984
13. Friedman J.H. Estimating functions of mixed ordinal and categorical variables using adaptive splines, Tech. Report LCS108, Department of Statistics, Stanford University, 1991a
14. Friedman J.H. Multivariate Adaptive Regression Splines (with discussion), *The Annals of Statistics*, Vol. 19, No. 1, 1991b, pp. 1-141
15. Friedman J.H. Fast MARS, Department of Statistics, Stanford University, Tech. Report LCS110, 1993
16. Golub G.H., Heath M., Wahba G. Generalized cross-validation as a method for choosing a good ridge parameter, *Technometrics*, Vol 21, 1979, pp. 215-223
17. Gutmann H.-M. A radial basis function method for global optimization, *Journal of Global Optimization*, Vol. 19, 2001, pp. 201-227
18. Hand D.J., Mannila H., Smyth P. Principles of data mining, MIT Press, 2001, 578 p.
19. Hardy R.L. Multiquadratic equations of topography and other irregular surfaces. *Journal of Geophysical Research*, Vol. 76, 1971, pp. 1905-1915
20. Hastie T., Tibshirani R., Friedman J. The elements of statistical learning: Data mining, inference and prediction, Springer, 2nd edition, 2009, 746 p.
21. Hocking R.R. Methods and Applications of Linear Models: Regression and the Analysis of Variance, Wiley-Interscience, 2003
22. Hurvich C.M., Tsai C-L. Regression and time series model selection in small samples, *Biometrika*, Vol. 76, 1989, pp. 297-307
23. Isaaks E.H., Srivastava R.M. An introduction to applied geostatistics. Oxford University Press, New York, USA, 1989
24. Ivakhnenko A.G. The Group Method of Data Handling - a rival of the method of stochastic approximation, *Soviet Automatic Control*, Vol. 3, 1968, pp. 43-71
25. Jekabsons G. Ensembling adaptively constructed polynomial regression models, *International Journal of Intelligent Systems and Technologies (IJIST)*, Vol. 3, No 2, 2008, pp. 56-61
26. Jekabsons G., Lavendels J. Polynomial regression modelling using adaptive construction of basis functions, *Proceedings of IADIS International Conference, Applied Computing 2008*, Algarve, Portugal, 2008, pp. 269-276
27. Jekabsons G. Adaptive Basis Function Construction: an approach for adaptive building of sparse polynomial regression models. *Machine Learning, In-Tech*, 2010, pp. 127-156.
28. Jin R., Chen W., Simpson T. Comparative studies of metamodelling techniques under multiple modelling criteria, *Journal of Structural and Multidisciplinary Optimization*, Vol. 23, No. 1, Springer, 2001 (13 pages)

29. Jin R., Chen W., Sudjianto A. On sequential sampling for global metamodeling in engineering design. Proceedings of ASME Design Engineering Technical Conferences and Computers and Information in Engineering Conference, Montreal, Canada, ASME, 2002, 10 p.
30. Kalnins K., Ozolins O., Jekabsons G. Metamodels in design of GFRP composite stiffened deck structure. Proceedings of 7th ASMO-UK/ISSMO International Conference on Engineering Design Optimization, Association for Structural and Multidisciplinary Optimization in the UK (ASMO-UK), ISBN: 978-0853162728, Bath, UK, 2008, pp. 263-273
31. Kennedy J., Eberhart R.C. Particle Swarm Optimization, Proceedings of the 1995 IEEE International Conference on Neural Networks, Perth, Australia, 1995, pp. 1942-1948
32. Kohavi R. A study of cross-validation and bootstrap for accuracy estimation and model selection, Proceedings of IJCAI-95, Mellish C.S. (ed.), Morgan Kaufmann, 1995, pp. 1137-1143
33. LeBlanc M., Tibshirani R. Combining Estimates in Regression and Classification, Technical Report 9318, Department of Statistics, University of Toronto, 1993
34. Lophaven S.N., Nielsen H.B., Sondergaard J. Aspects of the Matlab toolbox DACE, Technical report IMM-REP-2002-13, Technical University of Denmark, Denmark, 2002, 44 p.
35. Marler R.T., Arora J.S. Survey of Multi-Objective Optimization Methods for Engineering, Structural and Multidisciplinary Optimization, 26, 2004, pp. 369-395
36. Martin J.D., Simpson T.W. Use of Kriging models to approximate deterministic computer models, AIAA Journal, Vol. 43, No. 4, 2005, pp. 853-863
37. McDonald D.B., Grantham W.J., Tabor W.L., Murphy M.J., Response surface model development for global/local optimization using radial basis functions, Proceedings of the 8th Symposium on Multidisciplinary Analysis and Optimization, Long Beach, CA, 2000
38. Michalewicz Z. Genetic Algorithms, Numerical Optimization and Constraints, Proceedings of the 6th International Conference on Genetic Algorithms, Pittsburgh, 1995, pp. 151-158
39. Miller A. Subset selection in regression (2nd ed.), Chapman & Hall/CRC, 2002, 238 p.
40. Powell M.J.D. Radial basis functions for multivariable interpolation: a review, Algorithms for Approximation, Mason J.C., Cox M.G. (eds.), London, Oxford University Press, 1987
41. Press W.H., Teukolsky S.A., Vetterling W.T., Flannery B.P. Numerical Recipes, The Art of Scientific Computing, 3rd edition, Cambridge University Press, 2007
42. Pudil P., Novovicova J., Kittler J. Floating search methods in feature selection, Pattern Recognition Letters, Vol. 15, 1994, pp. 1119-1125
43. Queipo N.V., Haftka R.T., Shyy W., Goel T., Vaidyanathan R., Tucker P.K. Surrogate-based analysis and optimization. Progress in Aerospace Sciences, Vol. 41, Elsevier, 2005, pp. 1-28
44. Rissanen J. Modeling by shortest data description. Automatica, Vol. 14, 1978, pp. 465-471
45. Rojas F. J.E., Viana F.A.C., Rade D.A., Steffen Jr V. Force identification of mechanical systems by using particle swarm optimization, 10th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Albany, USA, 2004
46. Russell S.J., Norvig P. Artificial intelligence: a modern approach (2nd edition), Englewood Cliffs, New Jersey: Prentice Hall, 2002
47. Schwarz G. Estimating the dimension of a model. Annals of Statistics, Vol. 6, 1978, pp. 461-464
48. Shepard D. A Two-dimensional Interpolation Function for Irregularly-spaced Data, Proceedings of the 1968 23rd ACM National Conference, New York: ACM Press, 1968, pp. 517-524
49. Simpson T.W., Mauerty T.M., Korte J.J., Mistree F. Comparison of response surface and Kriging models for multidisciplinary design optimization, 7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis & Optimization, St. Louis, MO, USA, AIAA, Vol. 1, 1998, pp. 381-391
50. Webb A.R. Statistical pattern recognition (2nd ed), John Wiley & Sons 2002, 496 p.
51. Witten I.H., Frank E. Data mining: practical machine learning tools and techniques with Java implementations (2nd ed), SF, USA, Morgan Kaufmann, 2005
52. Wolpert D. Stacked generalization, Neural Networks, Vol. 5, 1992, pp. 241-259
53. Zhang P. Model selection via multifold cross validation. The Annals of Statistics, Vol. 21, 1993, pp. 299-313