Art of Surface Interpolation

Mgr. Miroslav Dressler, Ph.D.

Kunštát, 2009

Preface

The following text is based on my twenty years experience in developing a surface interpolation algorithm and five years experience with solving surface interpolation problems provided by people all over the world.

The success of interpolation and quality of the resulting surface depends on the configuration of input data, the selected method, parameters of interpolation, grid size and so on. From this point of view, surface interpolation can be considered as an art.

The first part describes mathematical elements of commonly used methods based on exact mathematical formulations such as linear combination of radial basis functions, statistical formulation of the best linear unbiased estimate or on the demand of minimal curvature of a resulting function.

The purpose of the second part is to design and implement a new interpolation method ABOS (Approximation Based On Smoothing), which would eliminate limitations of existing methods and which would be robust and flexible enough for interpolating any data set, such as a complex of geological and seismic measurements, temperature distribution, height of a snow layer, concentration of contaminants in an aquifer or digital model of terrain.

The new method is not based on a mathematical definition of a resulting interpolation function. Instead, it provides tools for modelling surface shapes – three types of numerical tensioning and smoothing – enabling to achieve smooth interpolation or approximation as well as an interpolation with sharp local extremes.

List of symbols

Symbol	***Meaning***
$\Re^3$	three dimensional Euclidean space
XYZ	sequence $\{(X_i, Y_i, Z_i) \in \Re^3,\ i=1,\dots,n\}$ of points in 3D space
$x1, x2$	minimum and maximum of x-coordinates of points *XYZ*
$y1, y2$	minimum and maximum of y-coordinates of points *XYZ*
$z1, z2$	minimum and maximum of z-coordinates of points *XYZ*
$f(x,y)$	interpolation / approximation function
D	planar domain of $f(x,y)$
$\gamma(h)$	variogram function
P	matrix representing grid values
$i1, j1$	size of the grid = number of columns and rows of the matrix *P*
Dx	step of the grid in the x-direction
Dy	the step of the grid in the y-direction
NB	matrix of the nearest points
K	matrix of distances
$K_{\max}$	maximal element of the matrix *K*
3D	three-dimensional
RS	resolution of map
Filter	parameter of the ABOS method used for setting resolution
Dmc	the minimal Chebyshev distance
$A \rightarrow B$	copy of the matrix (or vector or number) *A* into the matrix (or vector or number) *B*

Chapter 1

Introduction

Surface interpolation and construction of maps have been traditionally used in many fields such as physics, geophysics, geology, geodesy, hydrology, meteorology, bathymetry and so on.

The intention of this work is to compare commonly used interpolation / approximation methods, evaluate their advantages, insufficiencies and limitations and further to design and to implement a new universal interpolation / approximation method which would enable to solve a quite large class of tasks.

1.1 Formulation of the interpolation / approximation problem

Let us denote *XYZ* as a sequence $\{(X_i, Y_i, Z_i) \in \Re^3,\ i=1,\ldots,n\}$ of points in 3D space and *D* as a rectangular domain containing points $XY=\{(X_i, Y_i) \in \Re^2,\ i=1,\ldots,n\}$. In this work, the *solving an interpolation or approximation problem* will mean the finding a continuous function of two independent variables $f(x, y)$, for which $f(X_i, Y_i)=Z_i$ or $|f(X_i, Y_i)-Z_i|<\delta \quad \forall i=1,\ldots,n$ respectively.

Except for trivial cases (for example approximation by a plane or by a polynomial function of two independent variables of higher degree) it is usually not possible to express the interpolation / approximation function by a simple analytic formula. That is why the following procedure is used:

The domain *D* containing the points *XYZ* is covered by a regular rectangular grid. At each node of the grid the z-value is calculated / estimated by the method solving the above-mentioned problem using all points *XYZ* or using only the points *XYZ* belonging to the certain surrounding of the node. This procedure is called *gridding*.

The value of the function *f* can then be computed, for example, using the bilinear equation $f(x, y)=a \cdot xy+b \cdot x+c \cdot y+d$, where the coefficients *a*, *b*, *c* and *d* are determined by the corner points of the grid rectangle containing the point (x, y).

1.2 Commonly used approaches to solution

The goal of this section is to present commonly used techniques for solving interpolation / approximation problems and to evaluate their applicability for the solving practical tasks. The below presented interpolation / approximation methods are:

- ***Triangulation with linear interpolation***
- ***Natural neighbour***
- ***Inverse distance***
- ***Minimum curvature***
- ***Regression by plane with weights***
- ***Radial basis functions***
- ***Kriging***

Modification of these methods for solving large problems is described in the last paragraph of this chapter.

1.2.1 Triangulation with linear interpolation

The method of triangulation with linear interpolation is historically one of the first methods used before the intensive development of computers. It is based on the division of the domain *D* into triangles. Each triangle then defines, by its three vertices, a plane – that is why the resulting surface is per partes linear.

Advantages:

- very fast algorithm

- resulting surface is interpolative

Disadvantages:

- the domain of the function f is limited to the convex envelope of the points *XYZ*.

- resulting surface is not smooth and isolines consists of line segments

- the division into triangles may be ambiguous, as the following simple example of alternative division of rectangle shows – in the first case a valley was created, in the second case a ridge was created.

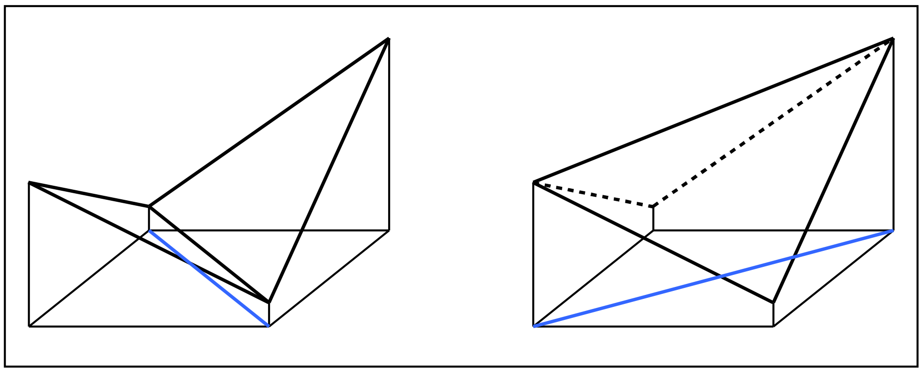


Fig. 1.2.1: Alternative division of rectangle into triangles.

Application:

This method is still used in geodesy and digital models of terrain. As a rule, characteristic points of terrain are measured – it means that the person performing terrain measurements surveys only points where the slope of terrain changes (tops, edges, valleys and so on) and thus avoids the above-mentioned ambiguity. For interpretation of such data, the Triangulation with linear interpolation method is quite suitable.

1.2.2 Natural neighbour

The Natural neighbor is an interpolation method based on Voronoi tessellation. Voronoi tessellation can be defined as "the partitioning of a plane with *n* points into *n* convex polygons such that each polygon contains exactly one point and every point in a given polygon is closer to its central point than to any other" (see [S11]). In other words, if $\{X_i\}_{i=1}^{n}$ is a given set of points in $\Re^2$, than the Voronoi polygon corresponding to the point X_i is the set $V_i = \{X \in \Re^2; |X, X_i| < |X, X_m| \quad \forall m \neq i\}$.

A description of the natural neighbor interpolation technique follows:
Given a set of data points distributed on a plane, natural neighbour interpolation computes the interpolated value for a given point *X* as the weighted sum of the points which are natural neighbors of *X*. The natural neighbors can be intuitively understood as those points which would be adjacent to *X* in a Voronoi tessellation of the point set including *X*.

Figure 1.2.2 depicts with black lines a Voronoi tessellation of the points *A*, *B*, *C*, *D* and *E*. The gray region marks the new Voronoi cell, which would be present if the point *X* were included in the tessellation. The weights of points *A*, *B*, *C*, *D* and *E* which are used to compute the interpolated value of *X* are respectively the areas of the grey region intersecting each original cell of *A*, *B*, *C*, *D* and *E* and are also known as the natural neighbor coordinates of *X*.

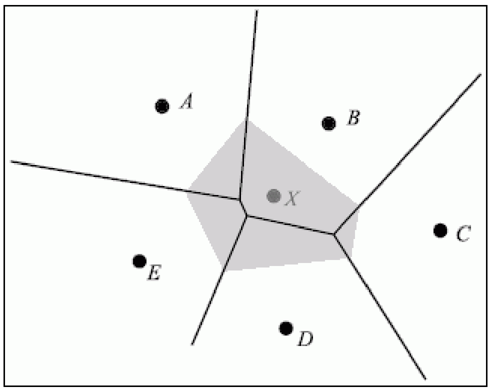


Fig. 1.2.2: New Voronoi cell and areas for computation of neighbor point weights.

The surface formed by natural neighbour interpolation has the useful properties of being continuous (C0) everywhere and passing exactly through z values of all data points. Moreover, the interpolated surface is continuously differentiable (C1) everywhere except at the data points, providing smooth interpolation in contrast to the Triangulation with linear interpolation method.

Advantages:

- fast algorithm

- resulting surface is interpolative and smooth except at the data points.

Disadvantages:

- the domain of the function f is limited to the convex envelope of the points *XYZ*

- the shape of the resulting surface is not acceptable in some fields such as in geology or hydrogeology.

Application:

The Natural neighbour method is mainly used in GIS systems as a digital model of terrain and fast interpolation of terrain data providing a smooth surface.

1.2.3 Inverse distance

This method computes a value of function *f* at an arbitrary point $(x, y) \in D$ as a weighted average of values Z_i:

$$f(x,y)=\sum_{i=1}^{n} Z_i w_i \text{, where } w_i=\frac{h_i}{\sum_{i=1}^{n} h_i} \text{, } h_i=\frac{1}{\sqrt{(x-X_i)^2+(y-Y_i)^2+\sigma^2}} \text{ and}$$

σ^2 is a smoothing parameter.

If the number of points n is too great, the value of $f(x,y)$ is calculated only from points belonging to the specified circle surrounding the point (x,y). The method was frequently implemented in the first stages of computers development.

Advantages:

- simple computer implementation; for its simplicity, the method is implemented in almost all gridding software packages

- if $\sigma^2=0$, the method provides interpolation.

Disadvantages:

- high computer time consumption if the number of points *n* is large (due to computation of distances)

- typical generation of "bull's-eyes" surrounding the position of point locations within the domain *D* – that is why the resulting function is not acceptable for most applications.

1.2.4 Minimum curvature method

This method and namely its computer implementation was developed by W.H.F. Smith and P.Wessel (see [1]) in 1990. The interpolated surface generated by the Minimum curvature method is analogous to a thin, linearly elastic plate passing through each of the data values with a minimum amount of bending. The algorithm of the Minimum curvature method is based on the numerical solution of the modified biharmonic differential equation $(1-T)\nabla^4 f(x,y)-(T)\nabla^2 f(x,y)=0$ with three boundary conditions:

1. $(1-T)\partial^2 f/\partial n^2+(T)\partial f/\partial n=0$
2. $\partial(\nabla^2 f)/\partial n=0$ on the edges
3. $\partial^2 f/\partial x\partial y=0$ at the corners

where

$T\in\langle 0,1\rangle$ is a tensioning parameter,

∇^2 is the Laplacian operator $\nabla^2 f=\partial^2 f/\partial x^2+\partial^2 f/\partial y^2$,

$\nabla^4=(\nabla^2)^2$ is the biharmonic operator

$\nabla^4 f=\partial^4 f/\partial x^4+\partial^4 f/\partial y^4+2\partial^4 f/\partial x^2\partial y^2$ and

n is the boundary normal.

If *T=0*, the biharmonic differential equation is solved; if *T=1*, the Laplace differential equation is solved – in this case the resulting surface may have local extremes only at points *XYZ*.

Advantages:

- speed of computation is high and an increasing number of points *XYZ* has small influence on decreasing the computational speed.

- suitable method for a large number of points *XYZ*.

Disadvantages:

- complicated algorithm and computer implementation

- if the parameter *T* is near zero, the resulting surface may have local extremes out of the points location

- bad ability to conserve extrapolation trends.

Application:

- universal method suitable for smooth approximation and interpolation (for example distribution of temperature, water heads, potential fields and so on).

1.2.5 Regression by plane with weights

This method is based on regression by plane $f(x,y)=ax+by+c$ using a weighted least square fit. The weight w_i assigned to the point (X_i,Y_i,Z_i) is computed as an inverse distance from point (x,y) to the point (X_i,Y_i). Then the minimum of the following function of three independent variables has to be found:

$F(a,b,c)=\sum_{i=1}^{n} w_i(f(X_i,Y_i)-Z_i)^2=\sum_{i=1}^{n} w_i(aX_i+bY_i+c-Z_i)^2$, which leads to a solution of three linear equations:

$$\frac{\partial F}{\partial a}=0=2\sum_{i=1}^{n} w_i X_i(aX_i+bY_i+c-Z_i)$$
$$\frac{\partial F}{\partial b}=0=2\sum_{i=1}^{n} w_i Y_i(aX_i+bY_i+c-Z_i)$$
$$\frac{\partial F}{\partial c}=0=2\sum_{i=1}^{n} w_i(aX_i+bY_i+c-Z_i)$$

After rearrangement the following equations are obtained:

$$a\sum_{i=1}^{n} w_i X_i^2+b\sum_{i=1}^{n} w_i X_i Y_i+c\sum_{i=1}^{n} w_i X_i=\sum_{i=1}^{n} w_i X_i Z_i$$
$$a\sum_{i=1}^{n} w_i X_i Y_i+b\sum_{i=1}^{n} w_i Y_i^2+c\sum_{i=1}^{n} w_i Y_i=\sum_{i=1}^{n} w_i Y_i Z_i$$
$$a\sum_{i=1}^{n} w_i X_i+b\sum_{i=1}^{n} w_i Y_i+c\sum_{i=1}^{n} w_i=\sum_{i=1}^{n} w_i Z_i$$

In addition to the regression by plane, some mapping packages, for example Surfer (see [S2]) or ConPac library (see [S10]), offer possibility to use polynomials of higher order.

Advantages:
- simple algorithm
- good extrapolation properties

Disadvantages:
- resulting function is only approximative
- slow speed of computation if *n* is great (due to computation of distances)

Application:
- surface reconstruction from digitized contour lines. The method was frequently used namely in the past, when contour maps were transferred from paper sheets to digital maps.

1.2.6 Radial basis functions

The method of Radial basis functions uses the interpolation function in the form:

$$f(x,y)=p(x,y)+\sum_{i=1}^{n} w_i\cdot\phi(|(x,y)-(X_i,Y_i)|) \qquad (1.2.6)$$

where

$p(x,y)$	is a polynomial
$w_i\in\Re$	are real weights
$\lvert(x,y)-(X_i,Y_i)\rvert$	is the Euclidean distance between points (x,y) and (X_i,Y_i)
$\phi(r)$	is a radial basis function

Commonly used radial basis functions are (c^2 is the smoothing parameter):

Multiquadric:	$\phi(r)=\sqrt{r^2+c^2}$
Multilog:	$\phi(r)=\log(r^2+c^2)$
Natural cubic spline:	$\phi(r)=(r^2+c^2)^{3/2}$
Natural plate spline:	$\phi(r)=(r^2+c^2)\cdot\log(r^2+c^2)$

The interpolation process starts with polynomial regression using the polynomial $p(x,y)$. Then the following system of n linear equations is solved for unknown weights w_i, $i = 1,...,n$:

$$Z_j-p(X_j,Y_j)=\sum_{i=1}^{n} w_i\cdot\phi(|(X_j,Y_j)-(X_i,Y_i)|),\ j=1,\dots,n$$

As soon as the weights w_i are determined, the z-value of the surface can be directly computed from equation (1.2.6) at any point $(x,y)\in D$.

Advantages:

- simple computer implementation; the system of linear equations has to be solved only once (in contrast to the Kriging method, where a system of linear equations must be solved for each grid node – see the next section)
- the resulting function is interpolative
- easy implementation of smoothing

Disadvantages:

- if the number of points n is large, the number of linear equations is also large; moreover the matrix of the system is not sparse, which leads to a long computational time and possibly to the propagation of rounding errors. That is why this method, as presented, is used for solving small problems with up to a few thousand points. Solving large problems is also possible, but requires an additional algorithm for searching points in the specified surrounding of each grid node – see paragraph ***1.2.8 Modification for solving large problems***.

Application:

- universal method suitable for use in any field

1.2.7 Kriging

Kriging is an interpolation method, which was originally developed for use in geology by D. G. Krige (see [2]), a professor at the University of Witwatersrand, South Africa, in the 1950s. In fact, the work of professor Krige is the base of science field called geostatistics.

Kriging is probably the most often used method for solving interpolation / approximation problems, namely because it is based on the statistical formulation of the *best linear unbiased estimate*. An important concept for deriving this method is *empirical* or *experimental variogram* $\gamma(h)$:

$$\gamma(h)=\frac{1}{2}\cdot\frac{1}{C(N(h))}\sum_{N(h)}(Z_i-Z_j)^2\text{, where}$$

$N(h)=\{i,j: |(X_i,Y_i)-(X_j,Y_j)|=h\}$ and $C(N(h))$ is the number of elements of the set $N(h)$.

For real data it is not probable that a pair of points will satisfy the condition $|(X_i,Y_i)-(X_j-Y_j)|=h$ and therefore for practical computation the set $N(h)$ is specified as $N(h)=\{i,j: |(X_i,Y_i)-(X_j,Y_j)|\in[h-\delta h, h+\delta h]\}$

The empirical variogram is approximated using the *theoretical variogram* or *model*. The commonly used models are:

Linear model:

$\gamma(h)=C_0+Sh,\ h\neq 0$ where C_0 is the so called nugget effect and S is the unknown slope.

Gaussian model:

$\gamma(h)=C_0+(C-C_0)\cdot\{1-\exp(-h^2/a^2)\}$

Exponential model:

$\gamma(h)=C_0+(C-C_0)\cdot\{1-\exp(-h/a)\}$

The Kriging method is intended for estimating the interpolation / approximation function $f(x,y)$ at point (X,Y) under the following assumptions:

a) The estimate Z of function $f(x,y)$ at an arbitrary selected point $(X,Y)\in D$ can be expressed as a weighted average $Z=\sum_{i=1}^{n} w_i\cdot Z_i$

b) Sum of weights is 1: $\sum_{i=1}^{n} w_i=1$

c) The estimate of value Z is unbiased i.e. the mean $E[f(x,y)-Z]=0$.

The weights are to be computed so that the dispersion variance $D(f(x,y)-Z)$ is minimal. In the next derivation we use f instead of $f(x,y)$. Taking into consideration the definition of dispersion variance $D(X)=E(X^2)-(E(X))^2$ and assumption c) it is obvious that we have to minimize the expression

$$E([f-\sum_{i=1}^{n} w_i\cdot Z_i]^2)=E([f\cdot(\sum_{i=1}^{n} w_i)-\sum_{i-1}^{n} w_i\cdot Z_i]^2)=E([\sum_{i=1}^{n} w_i\cdot f-\sum_{i=1}^{n} w_i\cdot Z_i]^2)=$$
$$=E([\sum_{i=1}^{n} w_i\cdot(f-Z_i)]^2)=\sum_{i=1}^{n}\sum_{j=1}^{n} w_i w_j E((f-Z_i)(f-Z_j))$$

From equation

$$E([Z_i-Z_j]^2)=E([Z_i-f+f-Z_j]^2)=E([Z_i-f]^2)-2E([(f-Z_i)(f-Z_j)])+E([f-Z_j]^2)$$

it follows that

$$E([(f-Z_i)(f-Z_j)])=\frac{1}{2}\cdot[E([f-Z_i]^2)+E([f-Z_j]^2)-E([Z_i-Z_j]^2)]\quad .$$

This term can be substituted into the expression which has to be minimized:

$$\sum_{i=1}^{n}\sum_{j=1}^{n} w_i w_j E((f-Z_i)(f-Z_j))=$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n} w_i w_j \frac{1}{2}[E([f-Z_i]^2)-E([f-Z_j]^2)-E([Z_i-Z_j]^2)]=$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n} w_i w_j \frac{1}{2}E([f-Z_j]^2)+\sum_{i=1}^{n}\sum_{j=1}^{n} w_i w_j \frac{1}{2}E([f-Z_i]^2)-\sum_{i=1}^{n}\sum_{j=1}^{n} w_i w_j \frac{1}{2}E([Z_i-Z_j]^2)$$

The first two terms can be simplified:

$$\sum_{i=1}^{n}\sum_{j=1}^{n} w_i w_j \frac{1}{2} E([f-Z_j]^2) = \sum_{j=1}^{n} \frac{1}{2} w_j E([f-Z_j]) \sum_{i=1}^{n} w_i = \sum_{j=1}^{n} \frac{1}{2} E([f-Z_j]^2)$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n} w_i w_j \frac{1}{2} E([f-Z_i]^2) = \sum_{i=1}^{n} \frac{1}{2} w_i E([f-Z_i]) \sum_{j=1}^{n} w_i = \sum_{i=1}^{n} \frac{1}{2} E([f-Z_i]^2)$$

The final form of the expression, which has to be minimized is:

$$2\sum_{j=1}^{n} \frac{1}{2} E([f-Z_j]^2) - \sum_{i=1}^{n}\sum_{j=1}^{n} w_i w_j \frac{1}{2} E([Z_i-Z_j]^2) = F(w_{1,}\ldots, w_n)$$

If the function $F(w_1,\ldots,w_n)$ has to be minimal, the partial derivatives according to w_k must be zero:

$$\partial F / \partial w_k = 0 \quad \forall k = 1,\ldots,n$$

It means that the following system of linear equations is obtained:

$$\sum_{i=1}^{n} \frac{1}{2} E([Z_k-Z_i]^2) w_i = \frac{1}{2} E([f-Z_k]^2), \quad k=1,\ldots,n$$

The halves of means $\frac{1}{2}E([Z_k - Z_i]^2)$ and $\frac{1}{2}E([f - Z_i]^2)$ can be substituted by values from the theoretical variogram. The system of equations must be completed by condition b), i.e. the sum of weights is 1 (that is why a fictive variable μ has to be added).
In the form of matrix notation the system of linear equations can be expressed as:

$$\begin{bmatrix} \Gamma & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} w \\ \mu \end{bmatrix} = \begin{bmatrix} \gamma^x \\ 1 \end{bmatrix}$$

where $\Gamma_{ij} = \gamma(|(X_i, Y_i) - (X_j, Y_j)|)$ and $\gamma_i^x = \gamma(|(x, y) - (X_i, Y_i)|),\ i, j = 1,\ldots, n$.

Advantages:

- the algorithm is based on statistical formulation of the best linear unbiased estimate which means, there is no better interpolation method from the statistical point of view.

Disadvantages:

- the weights w_i must be computed (i.e. the system of linear equations must be solved) for each node of the grid.
- if the number of points n is large, a large number of linear equations must be solved; that is why this method is used for solving small problems with up to a few thousand points. For solving large problems an additional algorithm for searching points in the specified surrounding of each grid node must be implemented, as described in the next paragraph.
- the method produces undesirable “pits” and “circular” isolines in the generated surface – see examples in paragraphs ***2.4.1 Smoothness of interpolation and oscillations, 2.4.2 Shape of generated surface*** and ***5.1 Zero-based maps***.

Application:

– universal method designed especially for geology and geophysics applications.

1.2.8 Modification for solving large problems

Except for the Triangulation with linear interpolation and Minimum curvature method, presented techniques need to compute horizontal distances from estimated point (grid node) to all points *XYZ*. For large problems (having thousands or more points) it means increasing of computational time, which brings necessity to reduce number of points involved into the estimation. This is solved by specification of the surrounding which determines the set of points included into the estimation. Of course, additional algorithm for searching of points falling into the surrounding must be implemented.

On one hand the method can be used for large problems, but on the other hand new complications are involved:

- The surrounding is usually circular (or elliptical, if an anisotropy has to be modelled) with specified radius. In addition, implementations of gridding procedures requires specification of maximal and minimal number of points, minimal and maximal number of points in each quadrant (octant) and so on. It means that additional parameters influencing the quality of resulting function has to be entered.
- The resulting surface may contain discontinuities (see [4]) as the set of selected points may change while moving the estimated point from one grid node to the next. This phenomenon is illustrated in the next chapter in the paragraph ***2.4.3 Conservation of an extrapolation trend***.
- In the case of the Radial basis functions method the system of linear equation must be completed and solved for each node of grid and the main advantage of this method is lost.

Chapter 2
The ABOS method

The goal of this chapter is to design an interpolation / approximation method which is sufficiently flexible and robust enough for solving large problems, provides results comparable with the Kriging method (respectively with the Radial basis function method or Minimum curvature method) and which does not have disadvantages and limitations of these methods presented in the first chapter.

The method was called ABOS (**A**pproximation **B**ased **O**n **S**moothing) and despite the fact that it should be used as an approximation method, according to its name, it can also be used for solving interpolation problems, as it will be explained in this chapter.

2.1 The definition of the interpolation function and notations

The interpolation function is determined by a matrix *P* of real numbers, whose elements (z-coordinates) are assigned to nodes of a regular rectangular grid covering the domain *D* (see the next figure).

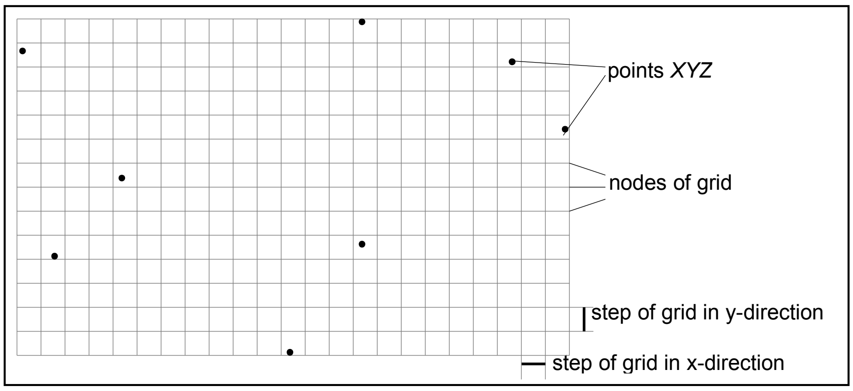


Figure 2.1: Regular rectangular grid for defining the interpolation function.

The value of the interpolation function at any point (x_0, y_0) within the grid can be evaluated from the equation of the bilinear polynomial $f(x,y)=a\cdot xy+b\cdot x+c\cdot y+d$, which is defined by coordinates of corner points of the grid rectangle containing the point (x_0, y_0).

The following notation is used in the next text:

$x1, x2$	minimum and maximum of x-coordinates of points *XYZ*
$y1, y2$	minimum and maximum of y-coordinates of points *XYZ*
$z1, z2$	minimum and maximum of z-coordinates of points *XYZ*
$i1, j1$	size of the grid = number of columns and rows of the matrix *P*
$P_{i,j}$	elements of the matrix $P,\ i=1,\dots,i1,\ j=1,\dots,j1$
DP	auxiliary matrix with the same size as the matrix *P*
Z	vector of z-coordinates of points *XYZ*
DZ	auxiliary vector with the same size as the vector *Z*

NB	matrix of the nearest points - integer matrix with the same size as the matrix P containing for each node of the grid the order index of the nearest point *XYZ*
K	matrix of distances – integer matrix with the same size as the matrix P containing for each node of the grid the distance to the nearest point *XYZ* measured in units of grid
$K_{\max}$	maximal element of the matrix *K*
Filter	a parameter of the ABOS method used for setting resolution
RS	resolution of map; $RS = max\{(x2-x1),(y2-y1)\}/Filter$
Dx	the step of the grid in the x-direction; $Dx = (x2-x1)/(i1-1)$
Dy	the step of the grid in the y-direction; $Dx = (y2-y1)/(j1-1)$
Dmc	the minimal Chebyshev distance between pairs of points *XYZ*; $Dmc = min\{max\{\lvert X_i - X_j\rvert, \lvert Y_i - Y_j\rvert\}\ ;\ i \neq j \wedge i, j = 1, \ldots, n\}$
A→B	means a copy of the matrix (or vector or number) *A* into the matrix (or vector or number) *B*

2.2 Interpolation algorithm

The algorithm of the ABOS method can be briefly described by the following scheme:

1. Filtering points *XYZ*, specification of the grid, computation of the matrices *NB* and *K*, *Z→DZ*, 0*→DP*
2. Per partes constant interpolation of values *DZ* into the matrix *P*
3. Tensioning and smoothing of the matrix *P*
4. *P+DP→P*
5. Z_i - $f(X_i, Y_i) \rightarrow DZ_i$
6. If the maximal difference $max\{DZ_i, i=1,\ldots,n\}$ does not exceed defined precision, the algorithm is finished
7. *P→DP*, continue from step 2 again (= start the next iteration cycle)

In the following paragraphs the particular steps of the algorithm are explained in full detail.

2.2.1 Filtering of points XYZ

If an interpolation / approximation problem has to be solved, it is necessary to take into consideration the fact that there may be some points *XYZ* with a horizontal distance less than the desired resolution of the resulting surface. That is why the first implemented algorithm in the ABOS method is the filtering of points *XYZ*. Filtering substitutes every two points (X_i, Y_i, Z_i), (X_j, Y_j, Z_j), such that

$$|X_i - X_j| < RS \ \wedge \ |Y_i - Y_j| < RS, \qquad (2.2.1)$$

by one point (X_k, Y_k, Z_k) with average coordinates i.e. $X_k = (X_i + X_j)/2$, $Y_k = (Y_i + Y_j)/2$ and $Z_k = (Z_i + Z_j)/2$.

The resolution *RS* is computed as $max\{(x2-x1),(y2-y1)\}/Filter$, where *Filter* is an optional parameter of the ABOS method. It is similar to the resolution of a digital picture – if the distance of two points with different colours is smaller than the pixel size of the digital picture, only one point with “average” colour can be seen.

The formulation of the filtering principle is easy, but computer implementation represents an efficiency problem, which is discussed in paragraph ***3.4.1 Implementation of filtering*** in ***Chapter 3***.

2.2.2 Specification of the grid

The size of the regular rectangular grid is set according to the following points:

1. The greater side of the rectangular domain D is selected, i.e. greater number of $x21=(x2-x1)$ and $y21=(y2-y1)$. Without loss of generality we can assume that $x21$ is greater.
2. The minimal grid size is computed as $i0=round(x21/Dmc)$, where Dmc is the minimal Chebyshev distance between pairs of points XYZ: $Dmc=min\{max\{|X_i-X_j|,|Y_i-Y_j|\}; i\neq j \wedge i,j=1,\dots,n\}$
3. The optimal grid size is set as: $i1=max\{k\cdot i0; k=1,\dots,5 \wedge k\cdot i0<Filter\}$
4. The second size of the grid is: $j1=round(y21/x21\cdot(i1-1))+1$

The presented procedure ensures that the difference between *Dx* and *Dy* is minimal i.e. the regular rectangular grid is as close to a square grid as possible.

2.2.3 Computation of matrices NB and K

The matrices *NB* and *K* are computed using the algorithm based on "circulation" around the points *XYZ*, as the following figure indicates:

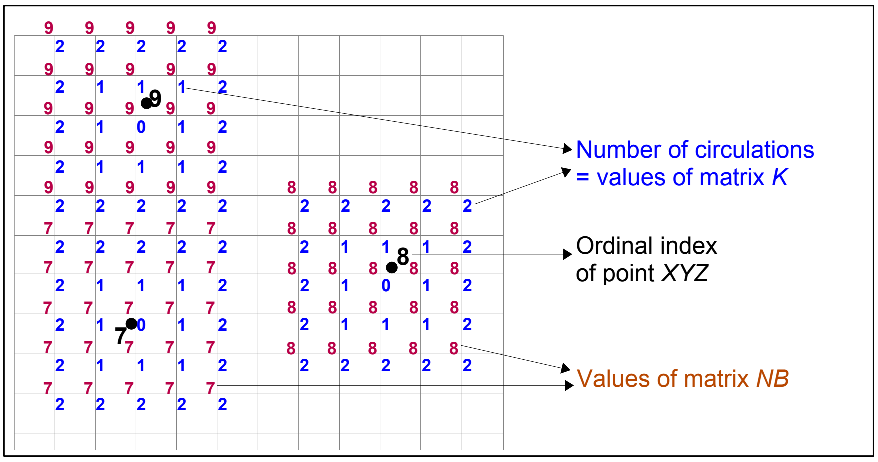


Fig. 2.2.3a: Computation of the matrices *NB* and *K*.

All elements of the matrices *NB* and *K* are initially set to zero and the process of circulation continues as long as there are zero values in the matrix *NB*. The Euclidean distance is compared only if the element $K_{i,j}$ corresponding to the evaluated node is not zero and $IC/\sqrt{2}\leq K_{i\,j}$, where IC is the ordinal number of the current circulation. By this way, the number of distance computations is significantly reduced.

The computation of the matrix *NB* defines a natural division of the domain of the interpolation function into polygons (so called Voronoi or Thiessen polygons, see the following figure), inside which interpolation with constant values is performed.

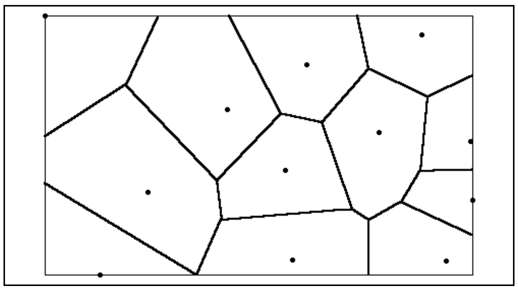


Fig. 2.2.3b: Division of the domain of the interpolation function.

2.2.4 Per partes constant interpolation

After computing the matrix of nearest points, per partes interpolation (see figure 2.2.4) is very simple: $P_{i,j} = DZ(NB_{i,j})$.

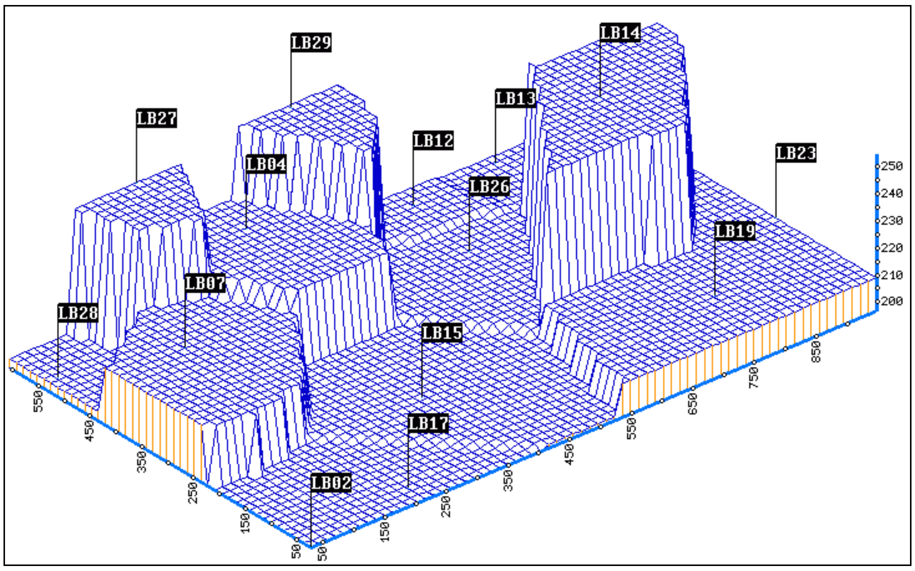


Fig. 2.2.4: Per partes constant interpolation.

2.2.5 Tensioning

Tensioning of the surface (see figure 2.2.5) modifies the matrix P according to the formula:

$$P_{i,j} = (P_{i+k,j} + P_{i,j+k} + P_{i-k,j} + P_{i,j-k})/4, \qquad (2.2.5)$$

where $k = K_{i,j}$, $i=1,\dots,i1$, $j=1,\dots,j1$

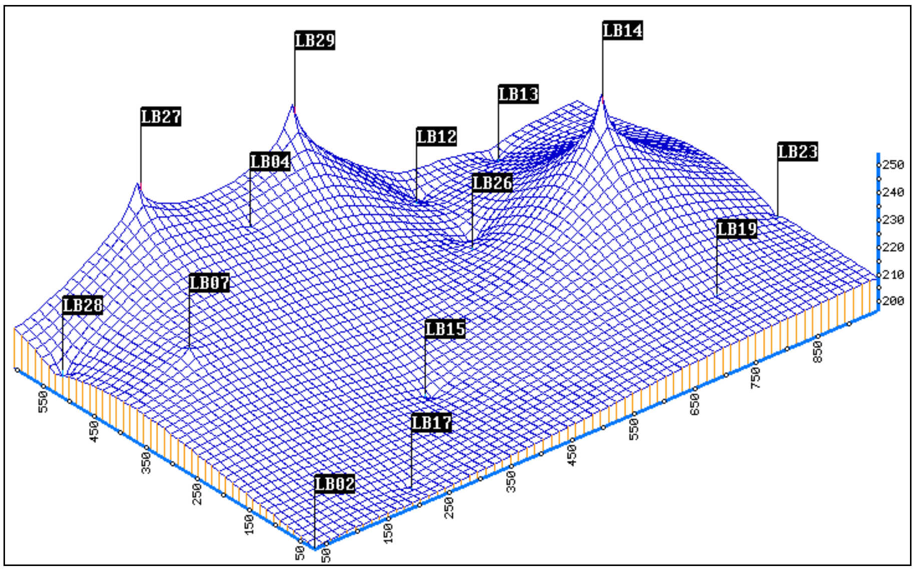
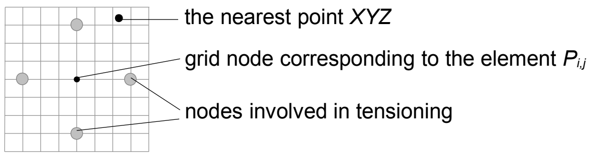


Fig. 2.2.5: Tensioning of the surface.

The following scheme shows the nodes (marked by grey circles) corresponding to the elements of the matrix *P*, which are involved in tensioning.


the nearest point *XYZ*
grid node corresponding to the element $P_{i,j}$
nodes involved in tensioning

Tensioning is repeatedly performed in the loop with this pattern:

```
DO N = MAX(4,Kmax/2+2),1,-1
  ...
ENDDO
```

If *k* is greater than decreasing loop variable *N*, then *k* = *N*.

2.2.6 Linear tensioning

Linear tensioning of the surface (see figure 2.2.6) modifies the matrix *P* according to the formula for weighted average:

$$P_{i,j}=(Q\cdot(P_{i+u,\,j+v}+P_{i-u,\,j-v})+P_{i-v,\,j+u}+P_{i+v,\,j-u})/(2\cdot Q+2) \qquad (2.2.6)$$
$$\forall\ i=1,\dots,i1,\ j=1,\dots,j1;\ K_{i,j}>0$$

where (u,v) is the vector from the node *i, j* to the nearest grid node of the point $NB_{i,j}$ and the weight $Q = L\cdot(K_{\max} - K_{i,j})^2$. The constant $L = 1/((0{,}107\cdot K_{\max} - 0{,}714)\cdot K_{\max})$ is an empirical constant suppressing the influence of $K_{\max}$.

In the implementation of the ABOS method there are four degrees of linear tensioning 0, 1, 2 and 3. Here presented formulas are valid for the default degree 1; their modifications for other degrees are described in the paragraph ***3.4.2 Degrees of linear tensioning***.

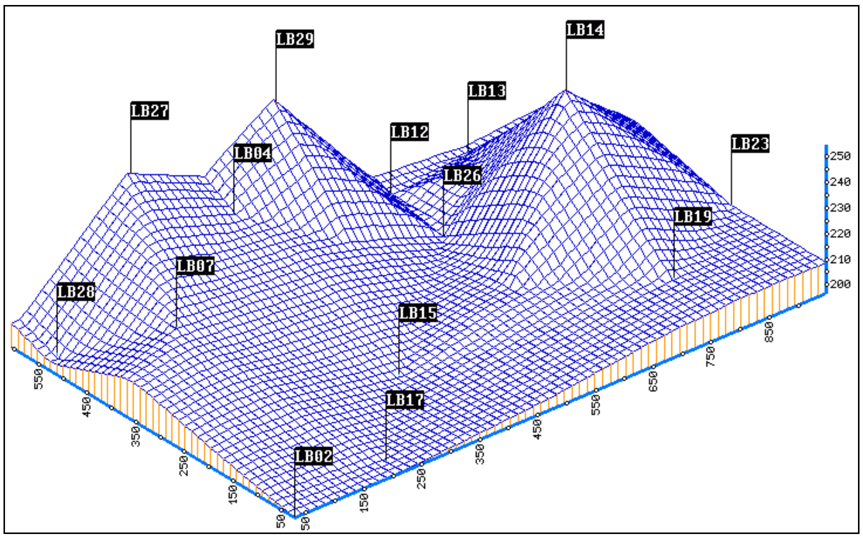
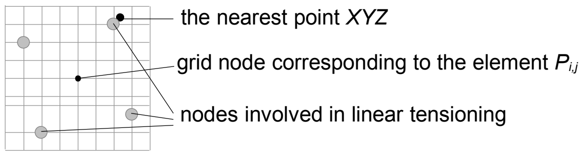


Fig. 2.2.6: Linear tensioning of the surface.

The following scheme shows the nodes (marked by grey circles) corresponding to the elements of the matrix P, which are involved in linear tensioning.


the nearest point *XYZ*
grid node corresponding to the element $P_{i,j}$
nodes involved in linear tensioning

Linear tensioning is repeatedly performed in the loop with this pattern:

```
DO N = MAX(4,Kmax/2+2),1,-1
  …
ENDDO
```

If the length $|(u,v)|$ of the vector (u,v) is greater than decreasing loop variable *N*, then the vector (u,v) is multiplied by constant c so that $c\cdot|(u,v)| = N$.

2.2.7 Smoothing

Smoothing (see figure 2.2.7) replaces elements of the matrix *P* by the value of weighted average:

$$P_{i,j}=(\sum_{k=i-1}^{i+1}\sum_{l=j-1}^{j+1}P_{k,l}+P_{i,j}\cdot(q\cdot t_{i,j}-1))/(q\cdot t_{i,j}+8),\ \ i=1,\dots,i1,\ j=1,\dots,j1 \qquad (2.2.7)$$

where q is the parameter of the ABOS method controlling smoothness of the interpolation / approximation (its default value is 0.5) and $t_{i,j}$ are weights, which are zero before the first smoothing and afterwards they are computed according to the formula

$$t_{i,j}=(\sum_{k=i-2}^{i+2}\sum_{l=j-2}^{j+2}(P_{i,j}-P_{k,l}))^2,\ \ i=1,\dots,i1,\ j=1,\dots,j1$$

and scaled into the interval *<0,100>*. In brief, it can be said, the values of $t_{i,j}$ are higher at nodes where the surface has a local extreme and lower at nodes where the surface is decreasing / increasing.

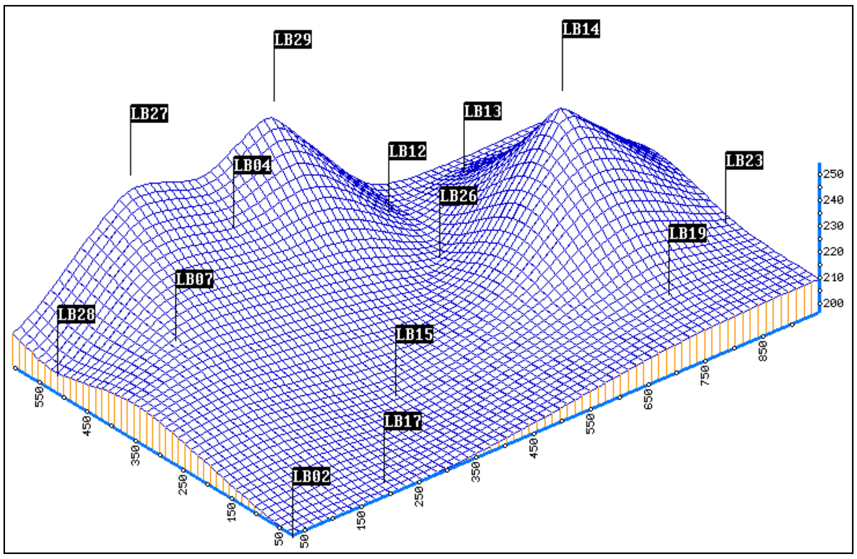
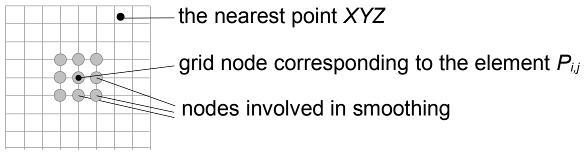


Fig. 2.2.7: Smoothing of the surface.

The following scheme shows the nodes (marked by grey circles) corresponding to the elements of the matrix *P*, which are involved in smoothing.


the nearest point *XYZ*
grid node corresponding to the element $P_{i,j}$
nodes involved in smoothing

Smoothing is repeatedly performed in the loop with this pattern:

```
DO N = MAX(4,Kmax*Kmax/16),1,-1
  …
ENDDO
```

As an option, the ABOS method enables to perform so called LES smoothing – in this case the formula (2.2.7) is not applied if the decreasing loop variable N is greater than $K_{i,j}+1$. This modification of smoothing suppresses oscillations and exceeding of local extremes, if they occur (see paragraph ***2.4.1 Smoothness of interpolation and oscillations***).

2.2.8 Iteration cycle

After smoothing, the matrices *P* and *DP* are added element by element and the result is stored again in the matrix *P*. Note that in the first iteration step the matrix *DP* is zero – that is why the matrix *P* does not change.

The tensioned and smoothed surface does not pass through the z-coordinates of points *XYZ* exactly, so the differences $DZ_i = Z_i - f(X_i, Y_i)$, $i = 1,...,n$ are calculated.

If the maximal difference $max\{DZ_i, i=1,\ldots,n\}$ is less than the specified accuracy of the interpolation / approximation multiplied by $(z2-z1)/100$, the algorithm of the ABOS method is finished. In the opposite case the matrix *P* is copied into the matrix *DP* and the algorithm continues from step 2, where a new iteration cycle begins. It differs from the first cycle only in these points:

- the matrix *DP* is not zero
- per partes constant interpolation is applied to the difference values DZ_i and not to the original z-coordinates of the points *XYZ*
- if the maximal difference in the current cycle is not less than the one in the previous cycle, the problem is considered to be non-converging and the algorithm is finished.

After each smoothing of the surface, the inaccuracy of the solution can be decreased by linear transformation of the matrix *P*:

$a \cdot P_{i,j}+b \rightarrow P_{i,j}$, where constants a and b minimize the term

$$\sum_{i=1}^{n} (a \cdot f(X_i, Y_i)+b-DZ_i)^2$$

In this way, the number of iteration cycles can be reduced by up to 30%.

The *accuracy*, another parameter of the ABOS method, is specified as a percentage value from the difference $z2-z1$. The default value is 1; if 0 is specified and the iteration process ends with zero maximal difference, then interpolation is achieved.

2.3 Flexibility of the ABOS method

Most interpolation methods offer some possibility how to modify a constructed surface. For example, the Radial basis function method enables to vary the smoothness of a surface using the smoothing parameter, the Kriging method may use different variogram models with different parameters and the Minimum curvature method uses the tension parameter.

The ABOS method can modify the created surface namely through the use of the smoothness parameter *q* (see paragraph ***2.2.7 Smoothing***) and an approximation can be achieved by setting an appropriate accuracy parameter. Moreover, the number of smoothing cycles suggested by the implementation of the ABOS method SURGEF can be increased (see ***3.10 Running SURGEF.EXE***) so that a suitable trend surface is obtained. The cross-section through seven surfaces in the next figure illustrates possible modifications of the surface shape generated by the ABOS method.

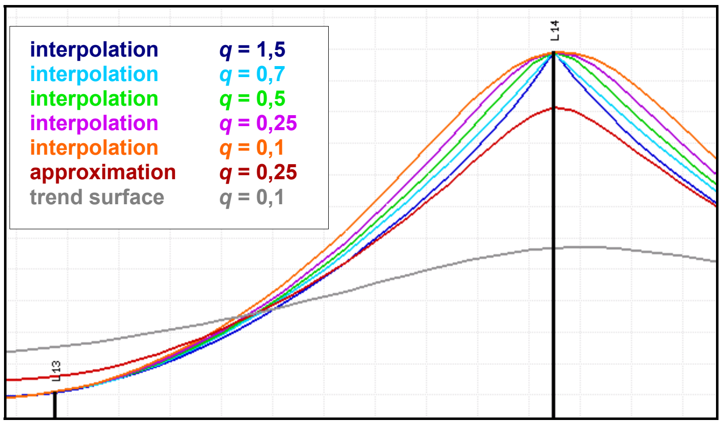


Fig. 2.3: Flexibility of the ABOS method interpolation / approximation.

The graphical user interface SurGe offers a tool for setting suitable parameters depending on the desired interpolation / approximation type (see paragraph ***4.2.3.2 Interpolation parameters***).

The usage of a trend surface is demonstrated in paragraph ***2.4.3 Conservation of an extrapolation trend*** and in section ***5.2 Extrapolation outside the XYZ points domain***.

2.4 Comparison with other interpolation methods

In this section the ABOS method is compared with three methods, which are considered to be the most significant:

- the *Minimum curvature* method with the tension 0.1
- the *Kriging* method using the linear model and zero nugget effect
- the *Radial basis functions* method using the multiquadric basis functions.

Although the Radial basis functions method was included, its graphical results are not presented because it provides almost the same result as the Kriging method (no differences can be seen in the graphical representation of results). Interpolations using Minimum curvature were performed using the program SURFACE, which is a part of the GMT system (see [S1]), while Kriging was performed using Surfer software (see [S2]).

To compare interpolation methods, we will use three data sets OSCIL, SHAPE and SIBIR. Characteristics of these examples are summarized in the following table:

Data set	***Number Of points***	***Domain range***		***Grid size***	***Grid step***		***Figure***
		x	***Y***		***X***	***Y***	
OSCIL	17	0-1	0-1	257x257	3.90625e-3	3.90625e-3	2.4.1a
SHAPE	30	0-1000	0-640	101x65	10	10	2.4.2a
SIBIR	13504	0-73600	0-51200	737x513	100	100	2.4.3a

The first example (see figure 2.4.1a) OSCIL is intended for examining of oscillation phenomena, which may occur especially if a smooth interpolation is used. The example SHAPE was designed for comparison of surface shape. Figure 2.4.2a shows the distribution of points *XYZ* and contains the horizontal projection of two cross-sections A-A' and B-B' used for detailed illustration of the surface shape. The third example (see figure 2.4.3a) is used for the comparison of trend conservation. The cross-section A-A' was designed for demonstrating extrapolation properties of the tested methods i.e. for evaluating how the tested methods conserve trend in areas where points are missing.

2.4.1 Smoothness of interpolation and oscillations

Depending on the configuration of the points *XYZ* a smooth interpolation may cause unwanted oscillations in the generated surface, which is a common problem of most smooth interpolation methods. In this paragraph we will examine oscillations on a specially designed example (see figure 2.4.1a) containing 13 points distributed along both diagonals of a square. All points have z-coordinate equal to zero except the point in the centre of the square, where the z-coordinate is one. We can expect that there will be oscillations between points having zero z-coordinates – that is why we will focus our attention to the cross-section going through points L09, L05 and L01.

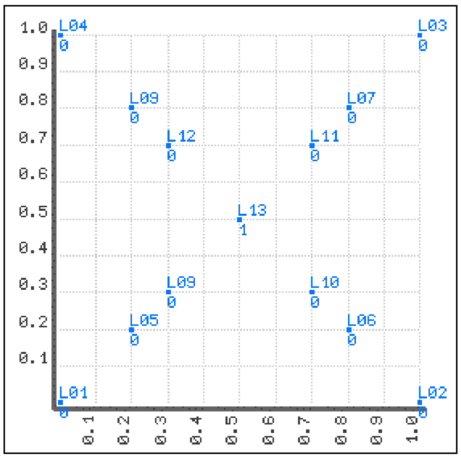


Fig. 2.4.1a: Distribution of points in the example OSCIL.

Firstly let us compare surfaces generated by the Minimum curvature, Kriging and ABOS method. It is obvious (see the next figure 2.4.1b) that the Minimum curvature method with tensioning 0.1 and the Kriging method with the linear model and zero nugget effect produce very similar surfaces. The ABOS method with a smoothing parameter of 9.0 produces a similar surface only along diagonals as indicates figure 2.4.1b and 2.4.1c, but otherwise there are apparent differences in the shape of undesirable “circular” contours (for example between points L05 and L06).

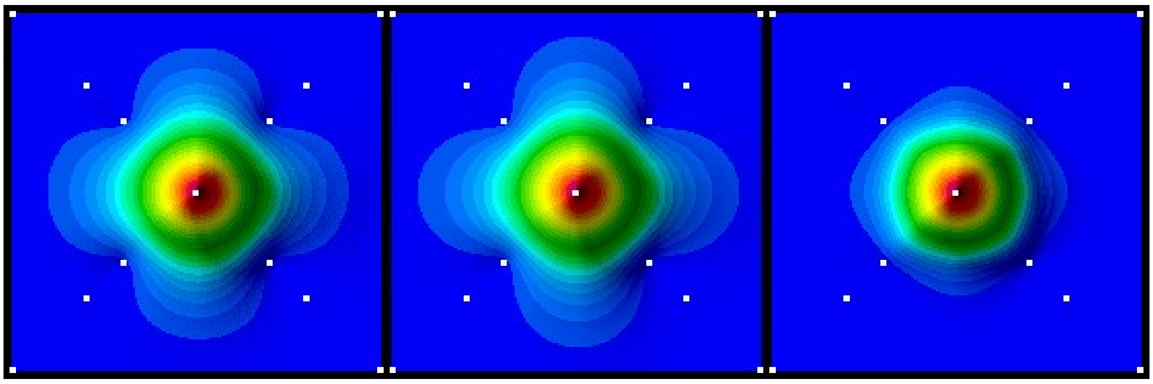


Fig. 2.4.1b: Surface generated by the Minimum curvature, Kriging and ABOS method, respectively.

In all three cases the tested interpolation methods create a sharp local extreme at the point L13 as follows from figure 2.4.1c.

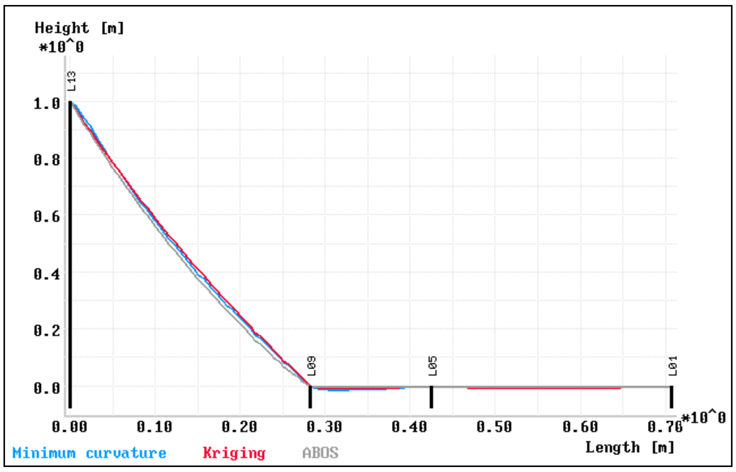


Fig. 2.4.1c: Cross-section through points L13, L09, L05 and L01.

Let's now examine details in the cross-section going through points L09, L05 and L01 where the oscillations were expected. As for the size of oscillation between points L09 and L05, the Kriging method provides the best result while the Minimum curvature method provides the worst result. As for oscillation between points L05 and L01, both named methods still produce oscillations but in the opposite order – the oscillation size produced by the Minimum curvature method is slightly smaller than in the case of Kriging method.

The ABOS method produced worse results than the Kriging and better results than Minimum curvature method between points L09 and L05, but oscillations between points L05 and L01 are negligible. Moreover, if LES smoothing is used during the interpolation process (see paragraph ***2.2.7 Smoothing***), the suppression of oscillations and improper extremes is very effective.

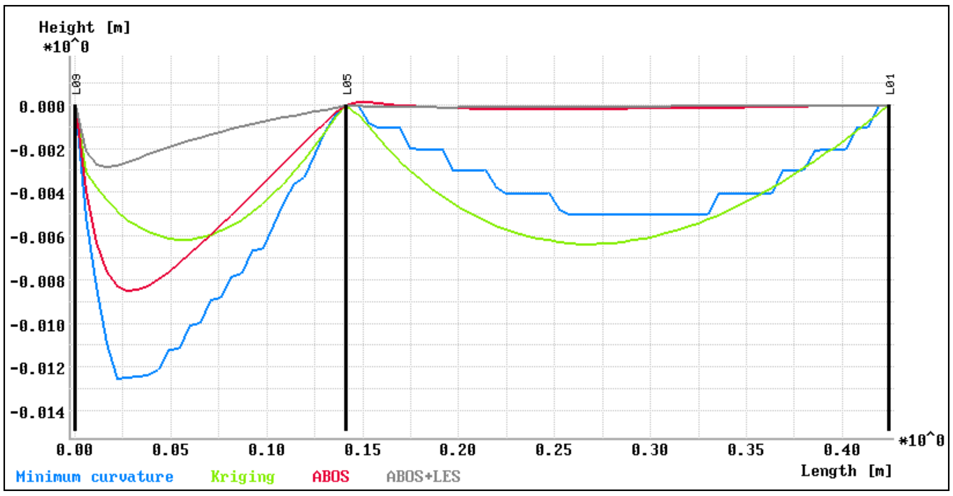


Fig. 2.4.1d: Cross-section through points L09, L05 and L01.

In any case, smooth interpolation may produce unwanted oscillations and improper extremes, which for example means that there are areas containing negative values in the solution of the interpolation problem, while only positive or zero values are possible. This situation is simply solved in the graphical user interface as described in paragraph ***5.1 Zero-based maps***.

2.4.2 Shape of generated surface

One of the first aspects evaluated by the user of interpolating technique is the shape of the created surface – the surface should be smooth, local extremes should be located at point positions, there should not be unsubstantiated bends on contour lines and so on.

For evaluation of these properties we will use data showed on figure 2.4.2a.

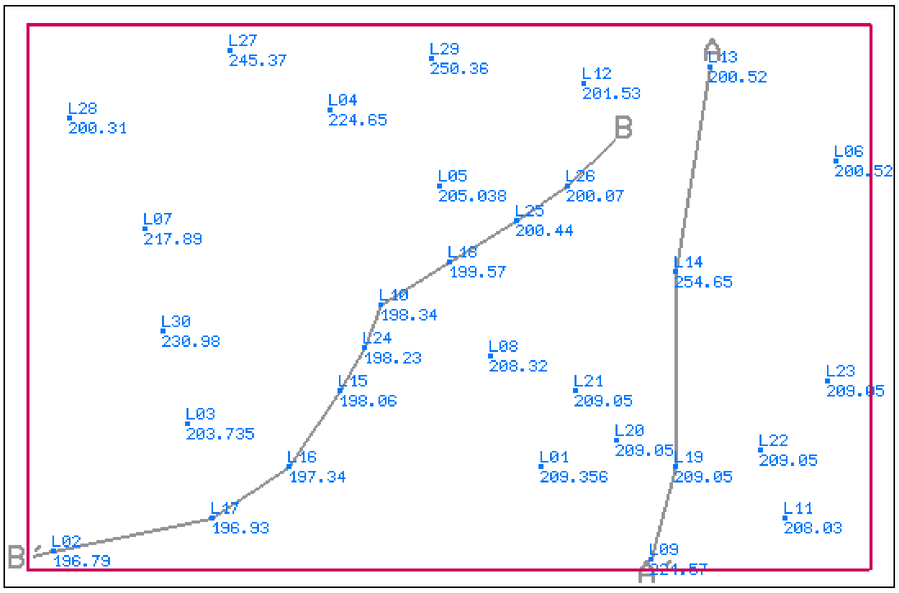


Fig. 2.4.2a: Distribution of points in the example SHAPE.

The next three figures show contours of surfaces created by the Minimum curvature method, Kriging and the ABOS method.

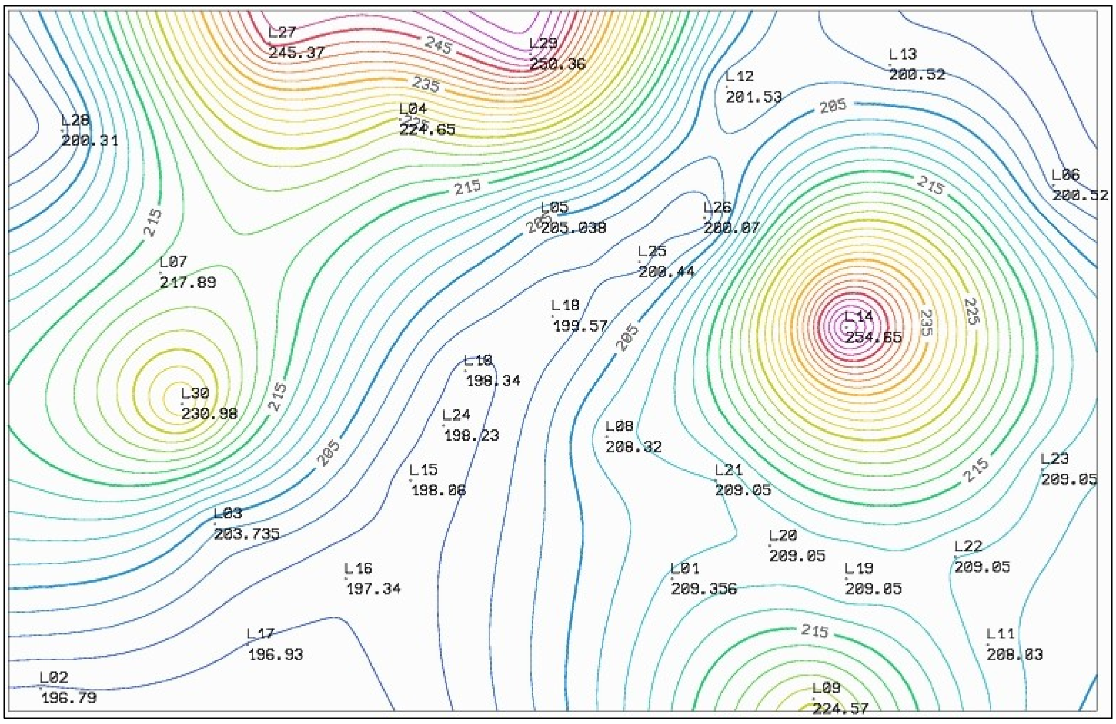


Fig. 2.4.2b: Interpolation of the SHAPE data set using the Minimum curvature method.

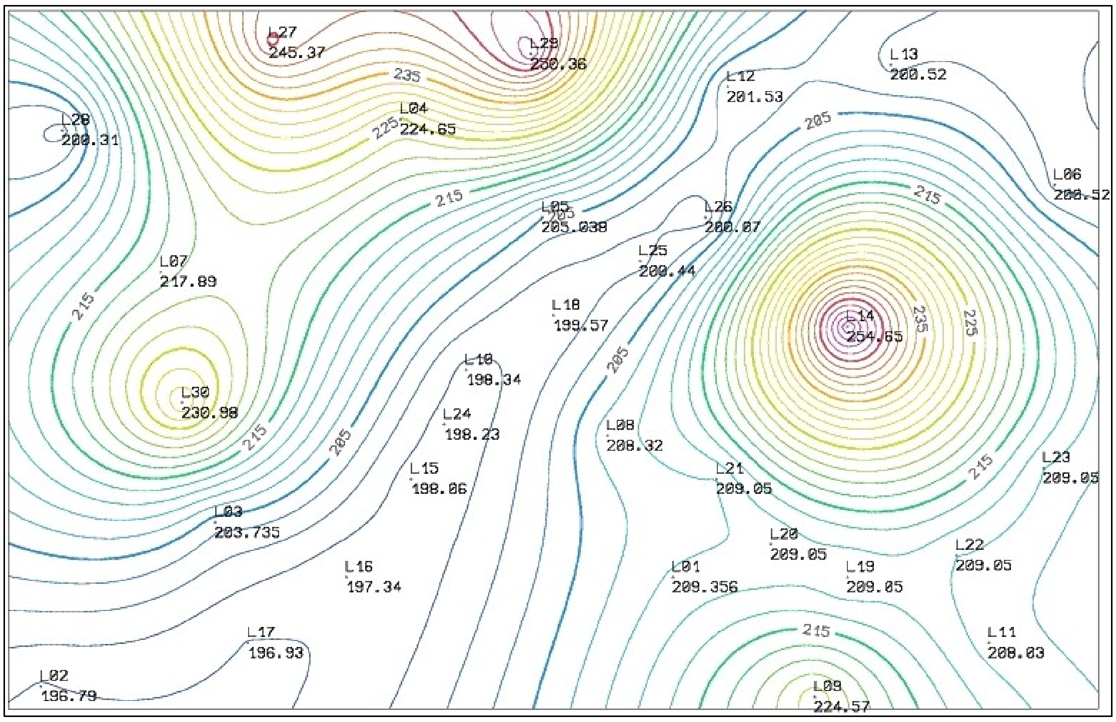


Fig. 2.4.2c: Interpolation of the SHAPE data set using the Kriging method.

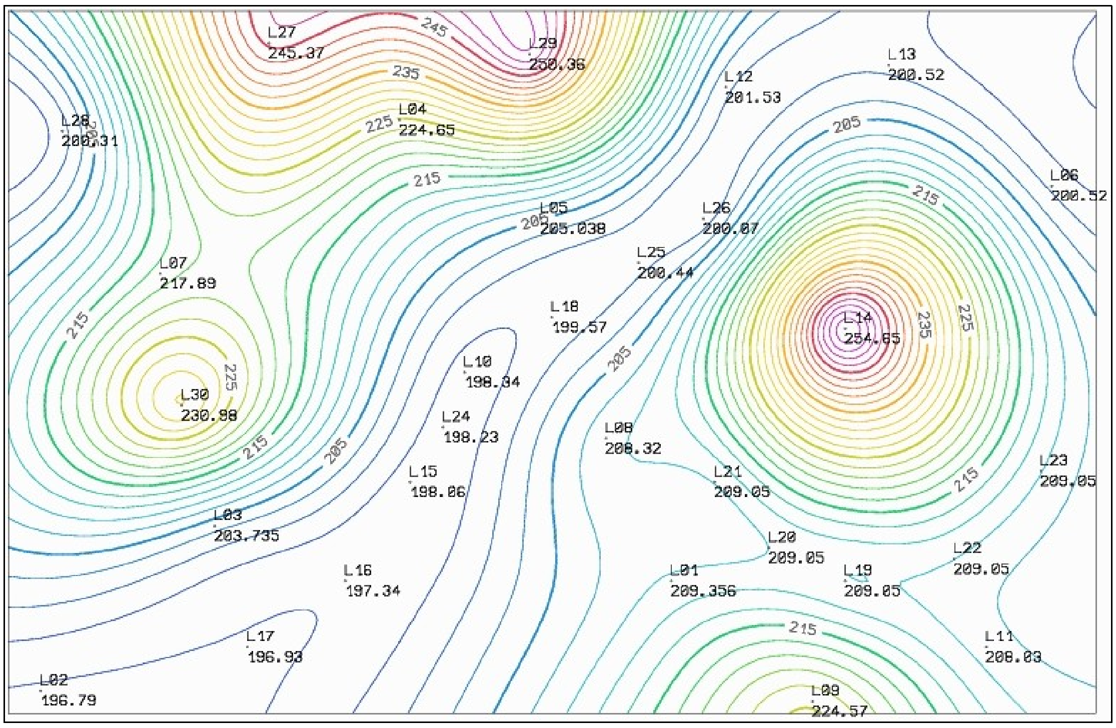


Fig. 2.4.2d: Interpolation of the SHAPE data set using the ABOS method.

The resulting function $f(x,y)$ can be considered as interpolative in all three cases – the maximal difference $max\{|f(X_i,Y_i)-Z_i|, i=1,\ldots,30\}$ does not exceed 0,002.

It is apparent that contours generated by the ABOS method are not so curved in the surrounding of some points (L18, L25, L26, L12, L13, L06, L23, L22) as in the case of the Kriging or Minimum curvature method.

Cross-sections provide another interesting comparison:

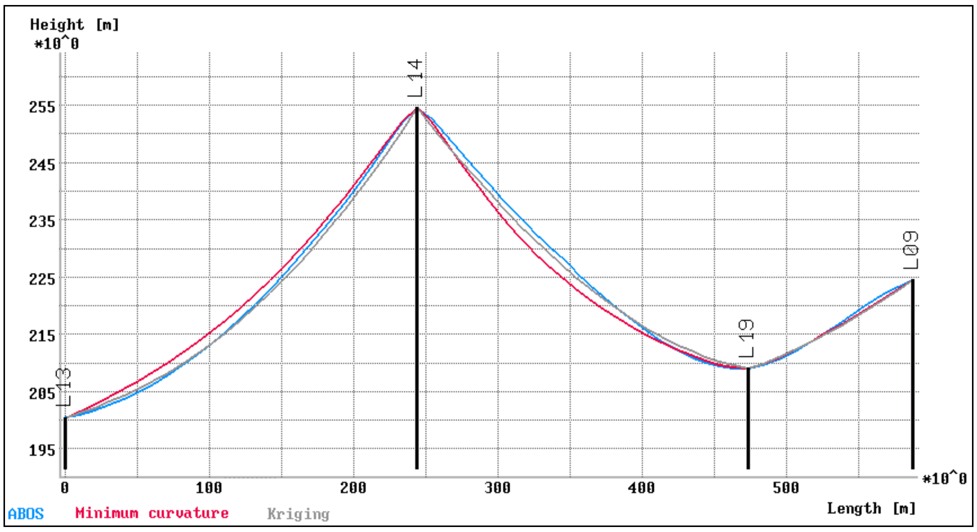


Fig. 2.4.2e: Cross-section A-A' through points L13, L14, L19 and L09.

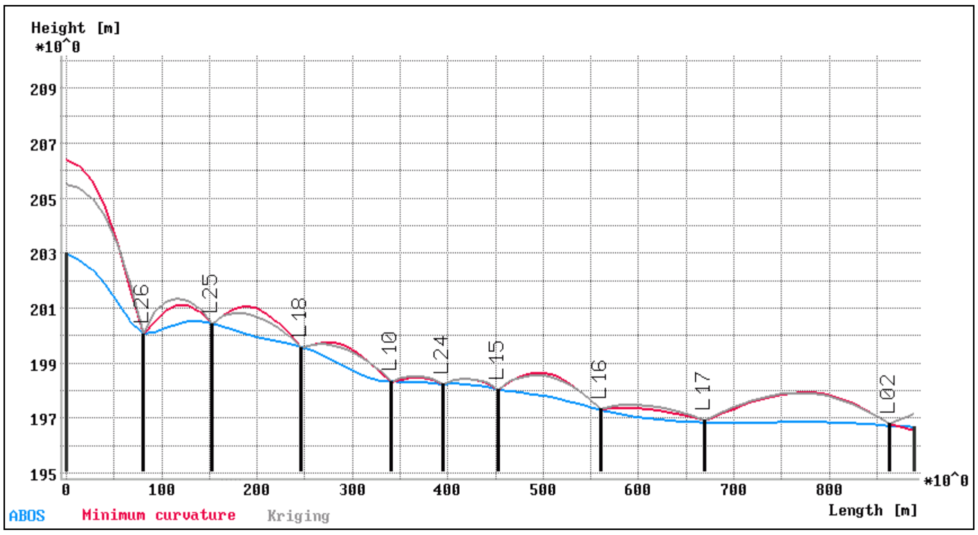


Fig. 2.4.2f: Cross-section B-B' through points L26, L25, L18, L10, L24, L15, L16, L17 and L02.

The cross-section A-A' (see figure 2.4.2e) shows that there is not significant difference between compared methods especially between the ABOS and Kriging method. Another situation is apparent in the case of cross-section B-B' leading through the "valley" (see figure 2.4.2f). The Minimum curvature method and Kriging produce "hills" between points although there is no reason for them.

2.4.3 Conservation of an extrapolation trend

The goal of the third example with the SIBIR data set (see the next figure) is to demonstrate the ability of tested methods to conserve trend in the regions without points.

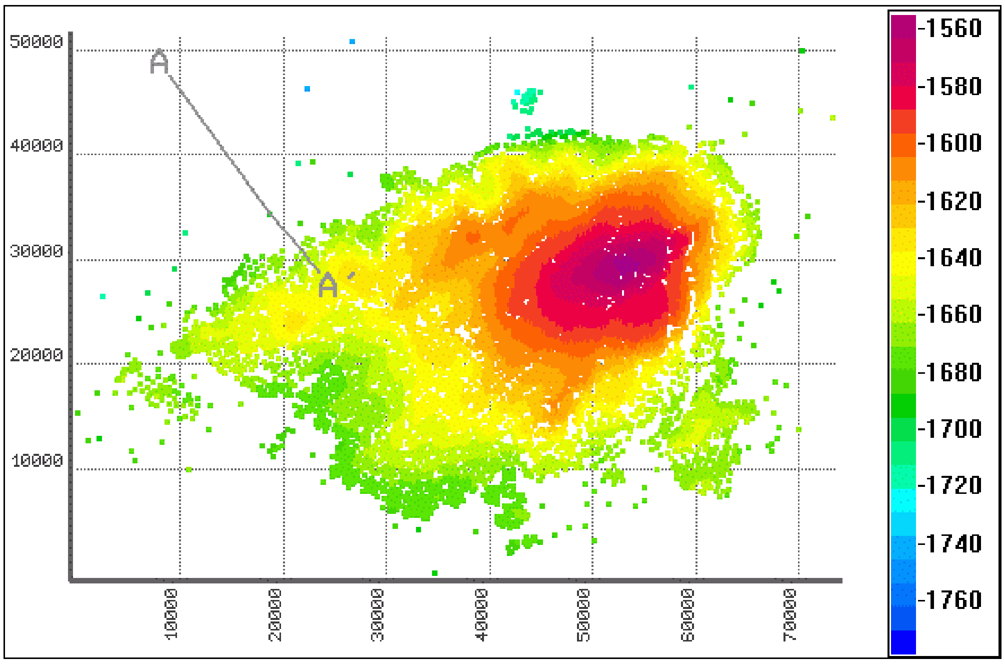


Fig. 2.4.3a: Distribution of points in the example SIBIR.

Again, the next three figures enable to compare the surface generated by the Minimum curvature, Kriging and ABOS methods. In the case of the ABOS method the interpolation with trend function was applied, which is the procedure implemented in the graphical user interface SurGe, as described in the paragraph ***4.2.3.10 Interpolation with a trend surface***.

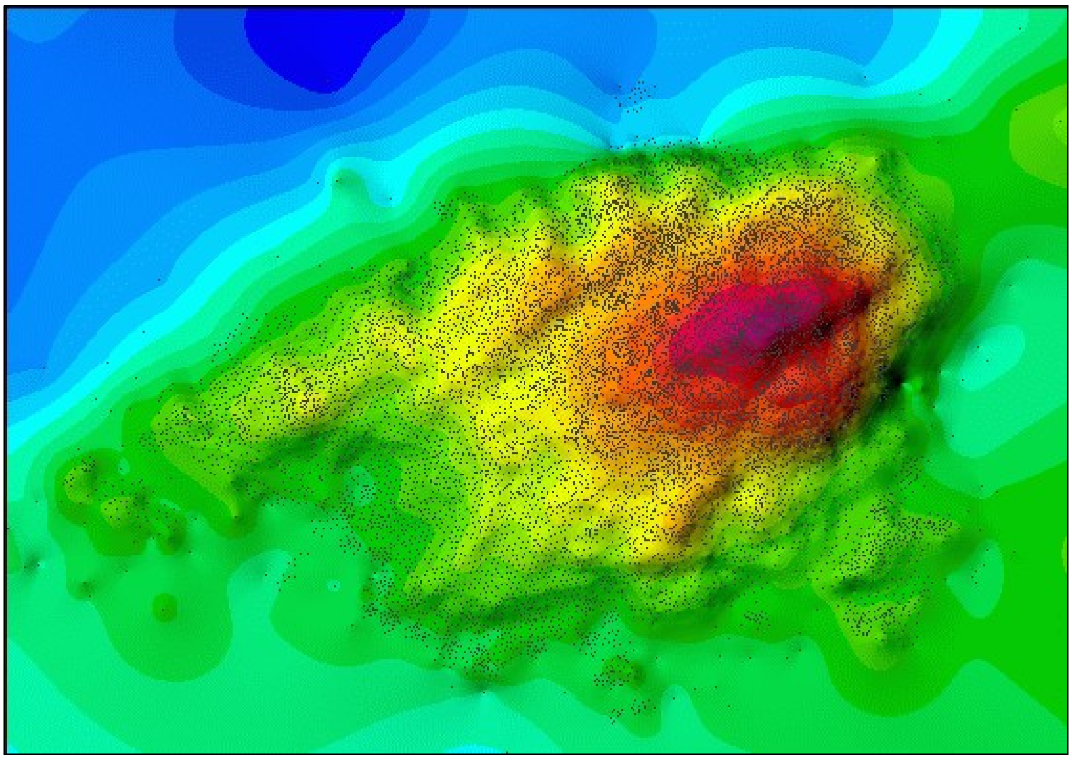


Fig. 2.4.3b: Interpolation of the SIBIR data set using the Minimum curvature method.

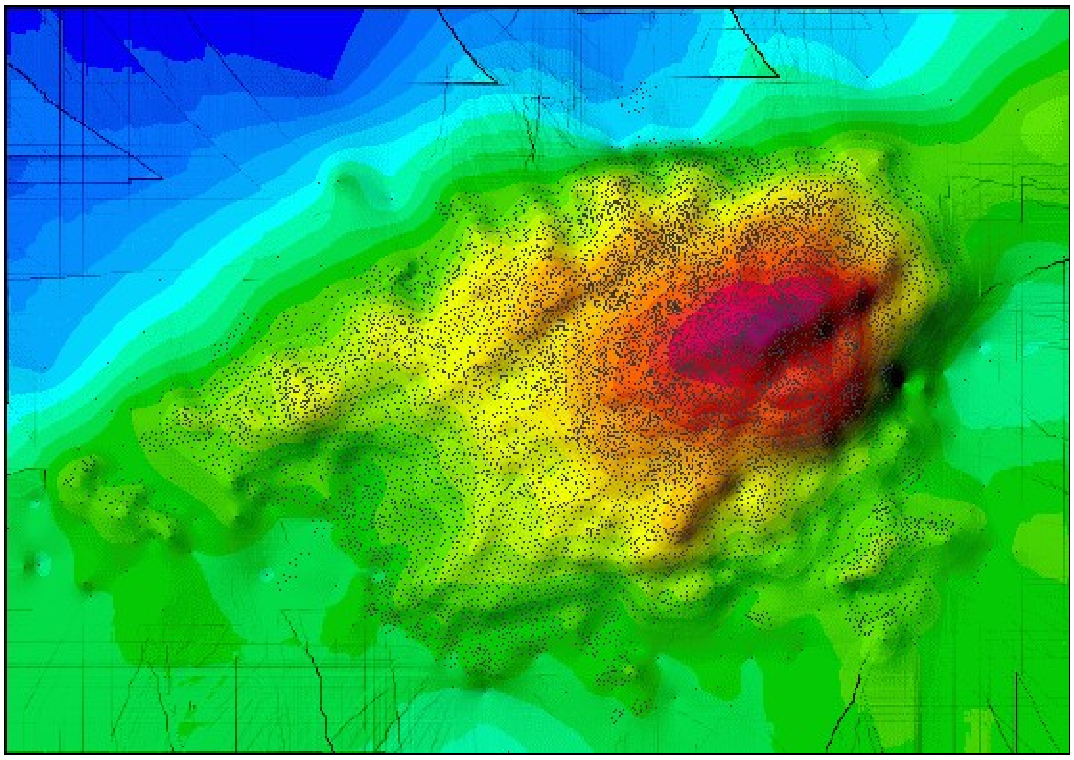


Fig. 2.4.3c: Interpolation of the SIBIR data set using the Kriging method.

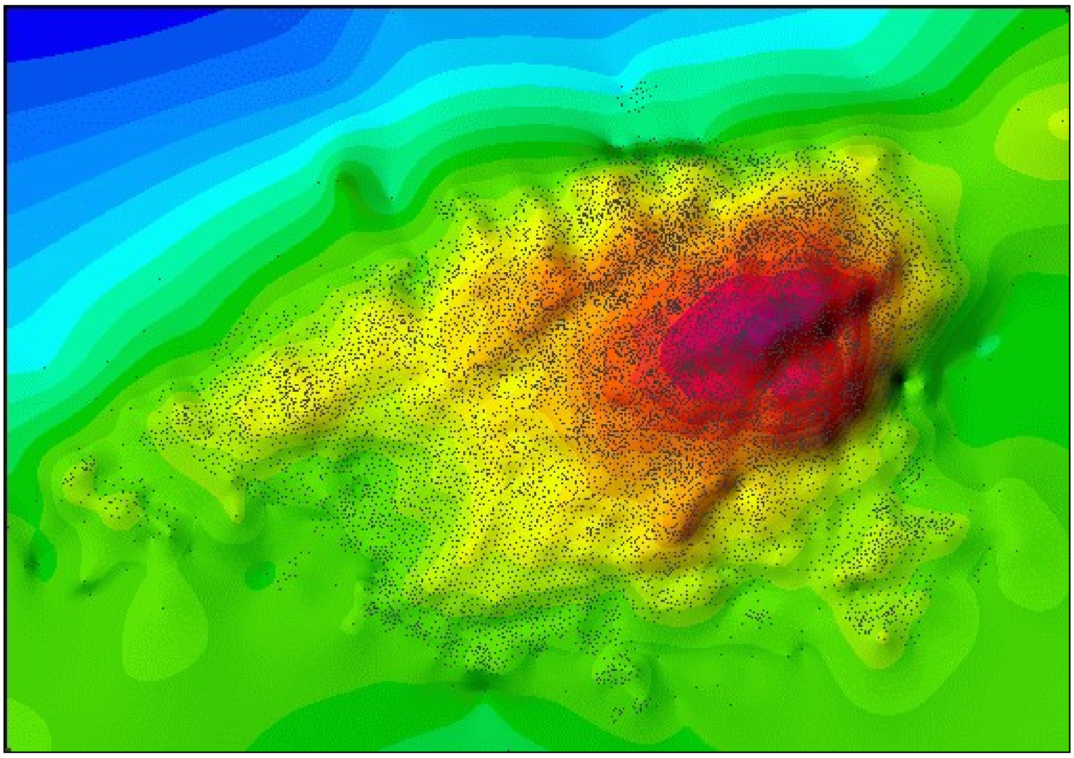


Fig. 2.4.3d: Interpolation of the SIBIR data set using the ABOS method.

All three methods generate apparently almost the same surface in the area densely covered by points *XYZ*, but at peripheral parts there are differences.

The surface created by the Minimum curvature method is smooth but it tends to be flat at the peripheral parts without points and even increases (see the upper right corner of the map in figure 2.4.3b) while it should decrease.

The Kriging method produces apparent “jumps” (discontinuities) at peripheral areas as expected due to the selection of points (max. 64) falling into the specified circle surrounding with a radius of 44600 units. The radius and maximal number of selected points was set by the used software – if one or both of these parameters were smaller, the jumps would be even higher. Without respect to the discontinuities, the ability of the Kriging method to conserve the extrapolation trend is better than in the case of the Minimum curvature method.

The surface created by the ABOS method does not have the deficiencies observed in the previous two cases and the cross-section A-A’ confirms that the conservation of extrapolation trend is the best (see the next figure).

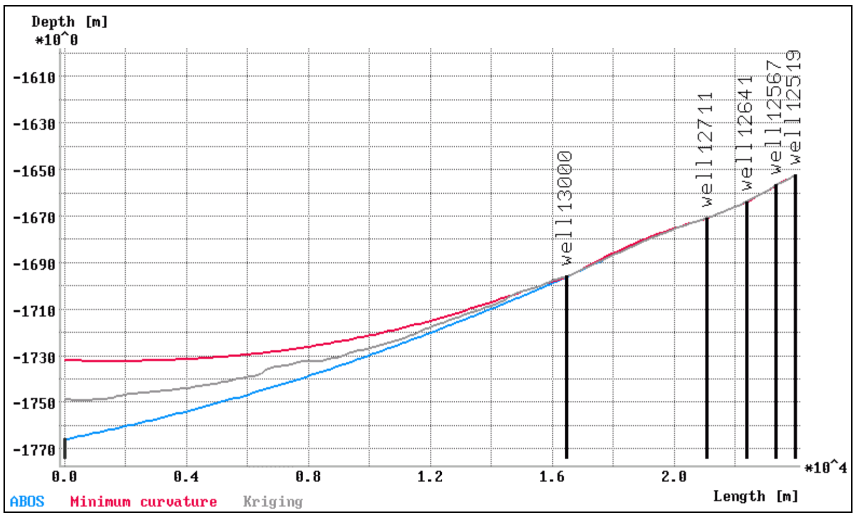


Fig. 2.4.2e: Cross-section A-A’.

2.4.4 Speed of interpolation

It has no sense to compare the speed of interpolation methods for small problems i.e. if the number of points is up to a few thousand, because all tested methods finish within a few seconds. However, if the number of points is greater, the computational speed becomes an important issue.

In the following list there are computational times consumed during solving interpolation with the SIBIR data set mentioned in the previous paragraph.

The Triangulation with linear interpolation	... 5 seconds
The Natural neighbour method	... 16 seconds
The Inverse distance method	... 350 seconds
The Radial basis functions method	... 515 seconds
The Minimum curvature method	... 11 seconds
The Kriging method	... 530 seconds
The ABOS method	... 18 seconds

The time was, of course, measured on the same computer with the CPU Intel Pentium M 1.6GHz and memory 512 MB RAM.

The time consumed by the Kriging method and the Radial basis functions method may seem to be curious, but with respect to the selected grid consisting of 737x513 nodes and the maximal number of selected points 64, the system of up to 64 linear equations had to be solved 378081 times.

2.4.5 Summary

The presented examples show that the ABOS method provides results comparable with highly sophisticated interpolation / approximation methods such as Minimum curvature, Kriging or Radial basis functions even if its mathematical basis is very poor. Moreover, ABOS is able to solve large problems without necessity to search points in specified surrounding of grid nodes and to solve systems of equations. As for computational speed, ABOS is about twenty times faster than Kriging or Radial basis function method.

Chapter 3
Computer implementation

The goal of the computer implementation of the ABOS method is the creation of a flexible program, which is suitable for usage in modern graphical computer applications.

This chapter describes important aspects of the SURGEF implementation while the graphical user interface SurGe for Windows operating system is described in the fourth chapter.

3.1 Selection of application type

Due to the fact that there is no uniform graphical user interface for all computer platforms, the program was designed as a console application named SURGEF with defined interface described in the documentation in full detail (see section ***3.9 Interface for user applications***.)

3.2 Selection of programming language

For the implementation of the ABOS method the programming language FORTRAN 77 was selected. The reasons for such a selection are:

1. Up to now, the programming language FORTRAN is the only language designed for the creation of scientific and technical applications.
2. In spite of the fact that the design of FORTRAN is obsolete, its development is continuing and its compilers exist for all computer platforms.
3. With respect to the simplicity of the language it is relatively easy to create highly optimised machine code.

Only a few functions are written in C language, namely the function for dynamical allocation of memory (this feature is missing in FORTRAN 77), filtering of input data (in order that the very fast sorting C function *qsort* could be utilized), reading of input data records (C library functions for input and output are faster than FORTRAN's) and computation of the convex envelope of points *XYZ* (the only non proprietary algorithm – see [S3]).

The source code was compiled with the GNU FORTRAN compiler *g77* and GNU C compiler *gcc*. Both compilers enable high optimisation both for machine code generation and for utilization of microprocessor architecture.

3.3 Program structure

3.3.1 Modularity

The program SURGEF implementing the ABOS method was designed as a set of modules performing individual parts of the solution. The groups of related modules are contained in these files:

SurgeF.f	main FORTRAN module together with initialisation subroutines
ReadDat.f	subroutines for data input and output
Nearest.f	subroutine for the computation of the matrix of nearest points and matrix of distances

Interp.f	subroutines for the interpolation
Common.f	common procedures and functions used in all other modules
CFunct.c	functions written in C language.

Although the memory needed for individual matrices is allocated as a one-dimensional array, all FORTRAN subroutines working with them use two-dimensional indexing, which is close to mathematical notation. Such trick is possible, because the arrays are passed into subroutines and functions by address and the FORTRAN compiler does not check compatibility between formal and actual parameters. This approach is commonly used in FORTRAN programs and enables to maintain and update the source code easily.

3.3.2 Memory allocation

As mentioned above, the memory for matrices and vectors is allocated dynamically, which is only possible using the C library function *malloc*. From this reason, the following C function callable from FORTRAN code was created:

```
// dynamical memory allocation for FORTRAN
void fmalloc_(int *mptr, int *nbytes)
{void *ptr; int amount; amount = *nbytes;
 if((ptr = malloc(amount)) == ((void *)0)) {*mptr=0; return;}
 *mptr = (int)ptr; return;}
```

The function **`fmalloc`** allocates a required amount of memory (**`nbytes`**) and assigns the pointer to the beginning of this memory into the variable **`mprt`**. If the memory cannot be allocated, a zero value is returned. The underscore after the function name is required for linker, because the *g77* compiler adds it after each subroutine or function name while the compiler *gcc* does not change function names.

In FORTRAN code the calling of **`fmalloc`** should look like:

```
CALL fmalloc(IAP,4*I1J1)      ! allocate memory for the matrix P
```

Here the **`IAP`** is a FORTRAN variable of type **`INTEGER*4`** containing the address of the allocated array after the calling of **`fmalloc`**. Then a subroutine declared for example as

```
SUBROUTINE SMOOTH(P)
REAL*4     P(I1,J1)
.
.
.
```

can be called by statement

```
CALL SMOOTH(%VAL(IAP))
```

where the **`%VAL(IAP)`** function returns the value contained by the variable **`IAP`**, but the subroutine **`SMOOTH`** considers it to be an address (FORTRAN assumes that parameters of subroutines and functions are passed only by address) which is all right, because **`IAP`** really contains an address assigned in the **`fmalloc`** function.

3.4 Description of selected algorithms

3.4.1 Implementation of filtering

As mentioned in section ***2.2.1 Filtering of points XYZ*** in the second chapter, filtering may represent an efficiency problem.

To test the condition (2.2.1) means to compare coordinates of all pairs of points *XYZ*. Such a problem is usually solved by nested loops with this pattern:

```
for (i=1;i<n;i++)
  {for (j=i+1;j<=n;j++)
    {compare coordinates of point i and j}
  }
```

It means that $n \cdot (n-1)/2$ comparisons are performed. If n is large, the computational time may be unacceptable. Taking into consideration the efficiency of today's computers, $n \cong 50000 - 100000$ is a critical value from this point of view. For example, filtering 100000 points using this algorithm took 170 seconds on the testing computer.

A significant increase in speed occurs when the points *XYZ* are sorted according to the x coordinates using a very fast standard C-library function *qsort* (that is why this filtering algorithm is one of the few algorithms written in C language). If the points *XYZ* are sorted according to the x coordinate, the above loops can be changed like this:

```
for (i=1;i<n;i++)
  {j=i+1;
   while ((j<=n)&&fabs(X[i]-X[j])<RS))
     {compare y-coordinate of point i and j; j++;}
  }
```

In other words, if the points *XYZ* are sorted according to the x-coordinates, there is no need to compare point i with all points $j=i+1,\ldots,n$ but only with the points j having $|X_i - X_j|$ less than the resolution. Using this approach the time needed for filtering 100000 points was reduced to 5 seconds.

An example of the filtering effect is displayed in the following figure. The input file contains 100000 points laying on a spiral and 50000 points forming a rotated square. These points are displayed in blue while data obtained after filtering are displayed in black. The *Filter* parameter was set to 100.

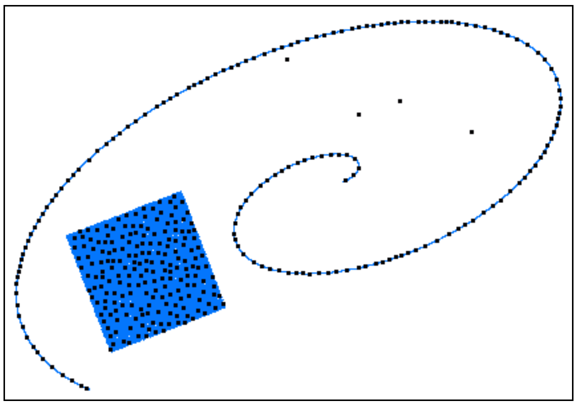


Fig. 3.4.1a: Filtered data.

It is obvious that filtering preserves the shape of clustered data while isolated points remain untouched.

According to performed tests, the above described filtering algorithm is effective for up to 300000 points – in such case the filtering process takes 20 seconds, which is still an acceptable time.

Another improvement can be achieved by implementing the so called super-block search strategy (see [3]), which consists of the following steps:

1. An ordinal number **IS[L],L=1,…,n** of the grid block is assigned to each point **L** (see the blue numbers in the next picture) using statements

```
I=(X[L]-X1)/Dx+1;
J=(Y[L]-Y1)/Dy+1;
IS[L]=I+(J-1)*I1;
```

2. Arrays **IS**, **X**, **Y** and **Z** are sorted according to values in the array **IS**.

3. An array **IN[I1*J1]** is set so that it contains the number of points belonging to grid block **K=1,..,I1*J1** and **IN[0]=0**. Then it is re-calculated (see the red numbers in the next picture) using the loop statement:

```
for (i=1;i<=I1*J1;i++) IN(i)=IN(i-1)+IN(i);
```

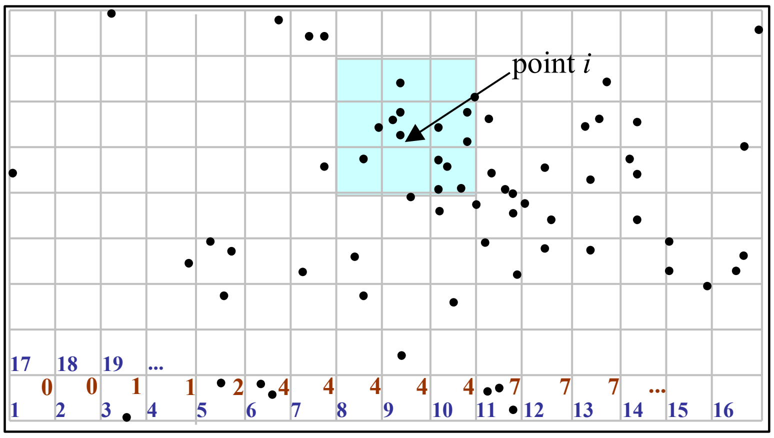
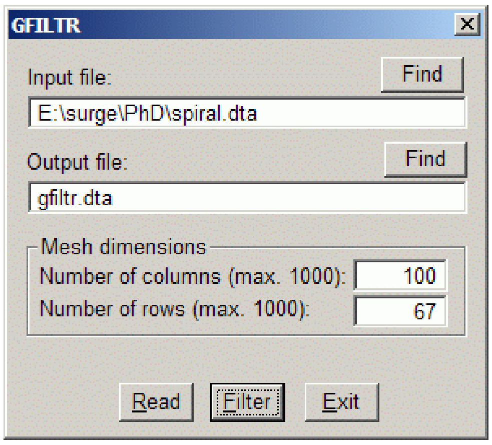


Fig. 3.4.1b: Super-block search strategy.

Now points within the grid block **K=1,..,I1*J1** can be indexed directly in the range **IN[K-1]+1,..,IN[K]** and during filtration we need to search only points in the block containing point *i* and in the eight adjacent blocks.

The super-block search strategy is the latest algorithm, which has been implemented in the SURGEF program and now it is being tested. Preliminary tests show that 300000 points can be filtered within 2 seconds, 1000000 points within 4 seconds and 5000000 points within 8 seconds.

The SurGe software package also contains another filtering algorithm implemented as a stand-alone utility GFILTR designed for pre-processing of a large amount of data.


GFILTR

Input file: E:\surge\PhD\spiral.dta

Output file: gfiltr.dta

Mesh dimensions

Number of columns (max. 1000):	100
Number of rows (max. 1000):	67

Fig. 4.3.1c: GFILTR utility for pre-processing of a large amount of data.

Filtering is performed in three steps:

1. Input data is read for the first time to set the minimal and maximal coordinates $x1$, $x2$, $y1$ and $y2$ of the domain D.
2. As parameters, the number of columns ($i1$) and rows ($j1$) of auxiliary regular rectangular mesh are specified. The size of the mesh blocks is calculated as $dx=(x2-x1)/(i1-1)$ and $dx=(y2-y1)/(j1-1)$. Four matrices XF, YF, ZF and WF with $i1$ columns and $j1$ rows are initialised to zero.
3. Input data is read for the second time. For each point (X_i, Y_i, Z_i) the following sequence of statements is performed:

```
i=round((Xi-x1)/dx)+1
j=round((Yi-y1)/dy)+1
w0=WF(i,j)
w1=v0+1
XF(i,j)=(w0*XF(i,j)+Xi)/w1
YF(i,j)=(w0*YF(i,j)+Yi)/w1
ZF(i,j)=(w0*ZF(i,j)+Zi)/w1
WF(i,j)=w1
```

Using this approach the elements $XF_{i,j}$, $YF_{i,j}$ and $ZF_{i,j}$ contain average coordinates of all points falling into the mesh block *i,j*. These coordinates are written into an output file only if the weight $WF_{i,j}>0$.

Figure 3.4.1d shows the result of the GFILTR utility applied for the above mentioned data example. Again, it is obvious that filtering preserves the shape of clustered data while isolated points remain untouched.

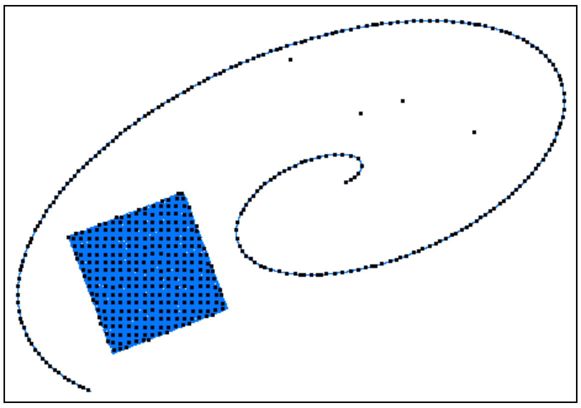


Fig. 3.4.1d: Data filtered by GFILTR utility.

As for efficiency, the GFILTR utility filters 300000 points within 5 seconds and 5000000 points within 40 seconds.

3.4.2 Degrees of linear tensioning

There are four degrees of linear tensioning (0-3) implemented in SURGEF. The formula for linear tensioning (2.2.6) can be expressed in this generalized form:

$$P_{i,j}=(Q\cdot(P_{i+u,j+v}+P_{i-u,j-v})+R\cdot(P_{i-v,j+u}+P_{i+v,j-u}))/(2\cdot Q+2\cdot R)$$
$$\forall i=1,\ldots,i1,\ j=1,\ldots,j1;\ K_{i,j}>0$$

where the weights Q and R, depending on the degree of linear tensioning, are calculated as follows:

Degree	Q	R	L
0	$L \cdot (K_{\max} - K_{i,j})^2$	1	$0{,}7/((0{,}107 \cdot K_{\max} - 0{,}714) \cdot K_{\max})$
1	$L \cdot (K_{\max} - K_{i,j})^2$	1	$1{,}0/((0{,}107 \cdot K_{\max} - 0{,}714) \cdot K_{\max})$
2	$L \cdot (K_{\max} - K_{i,j})$	1	$1{,}0/(0{,}0360625 \cdot K_{\max} + 0{,}192)$
3	1	0	-

Formulas for the computation of the constant L are empirical and their role is to suppress the influence of $K_{\max}$.

The figure 3.4.2 contains a cross-section plot demonstrating the typical influence of the linear tensioning degree.

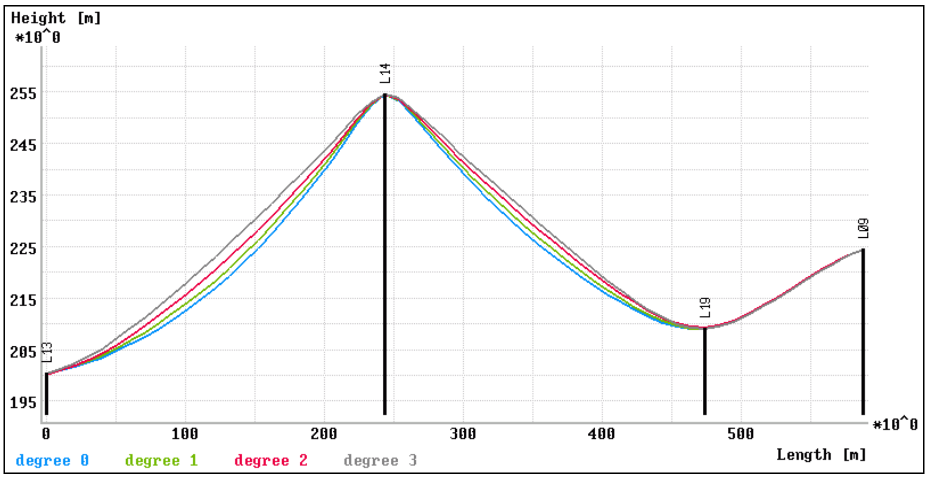


Fig. 3.4.2: Influence of the linear tensioning degree.

3.4.3 Smoothing and tensioning on grid boundary

The formulas for tensioning (2.2.5) and (2.2.6) and smoothing (2.2.7) are in fact the formulas for the computation of weighted average. For example in the case of smoothing, z-coordinates at 9 nodes of the grid are included in the weighted average; however, on the grid boundary only 6 or 4 nodes are available for smoothing (see figure 3.4.3a), which has an undesirable influence on the generated surface – the contours tend to be perpendicular to the grid boundary.

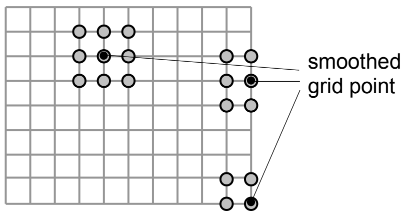


Fig. 3.4.3a: Nodes included in smoothing

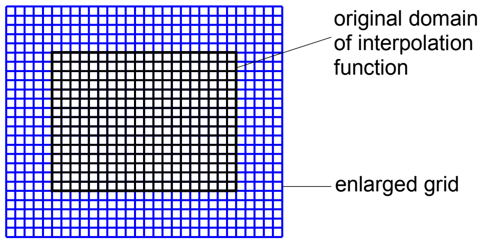


Fig.3.4.3b: Enlargement of grid

To suppress this phenomenon, SURGEF uses an enlarged grid. This grid enlargement is specified as a number of additional columns and rows symmetrically exceeding the original domain of the interpolation function – see blue lines in figure 3.4.3b where the grid size enlargement is 5.

3.5 Data compatibility with other systems

After thorough examination of other mapping and gridding software, it was decided to keep primary compatibility of input / output data formats with the Surfer software (see [S2]), because the majority of related software uses Surfer data format either directly or supports its import and export.

Namely it means, the SURGEF program reads points *XYZ* from the ASCII files in Surfer format and is also able to create grids as ASCII files, which are compatible with Surfer grids (see section ***3.8 Format of input and output files***).

Moreover the graphical interface SurGe for the SURGEF program supports a lot of other commonly used map formats, as described in section ***4.3 Supported map formats***.

3.6 Map objects

In addition to points *XYZ*, the ABOS implementation supports other objects used for the definition of maps:

- *Added points* are the *XYZ* points added by the user in order to modify the shape of the resulting surface according to his / her concept.
- *Spatial polylines* are 3D polylines which are involved in the interpolation / approximation process; they can be used namely for the settings of the boundary conditions.
- *Boundary* is one or more polylines in the horizontal plane intended for the definition of the interpolation / approximation function domain.
- *Faults* are lines along which the resulting surface has to be discontinuous.

The implementation of the map objects is explained in the following paragraphs.

3.6.1 Added points

Added points are treated in the same manner as the points *XYZ*; in other words, they are simply added to the sequence of the points *XYZ*.

3.6.2. Spatial polylines

A spatial polyline is defined by the x, y and z coordinates at each of its vertex point. In fact, SURGEF does not work with polylines directly – it works only with the points, which are evenly distributed along the polylines. The number of evenly distributed points is specified as a polyline parameter.

3.6.3 Boundary

A boundary is handled as a horizontal polyline. Its role is to define the domain of the interpolation / approximation function. If there is no boundary in the input data, the size of the domain (rectangular area) is given by the minimal and maximal coordinates of *XYZ* points. This size can be changed by a boundary – if a boundary exists and if it is involved in the interpolation, the size of the rectangular area is given by the minimal and maximal coordinates of the boundary points. The example of boundary use is in sections ***5.2 Extrapolation outside the XYZ points domain*** and ***5.6 Digital model of terrain***.

3.6.4 Faults

A fault is a sequence of line segments (in the horizontal plane), at which the resulting surface has to be discontinuous. The line segment of a fault is defined by the pair of points having specified x and y coordinates at each end.

The existence of faults affects the computation of the matrix *NB*, *K* and *P* according to the following rules (see the next figure):

- Elements of the matrix *P* corresponding to the nodes near the fault are not defined.
- Undefined nodes are involved in the computation of the matrix *K* as if they were points *XYZ*.
- The ordinal number of the nearest point *XYZ* is assigned to the element of the matrix *NB* only if the point is not on the opposite side of the fault.

the points *XYZ*, as the following figure indicates:

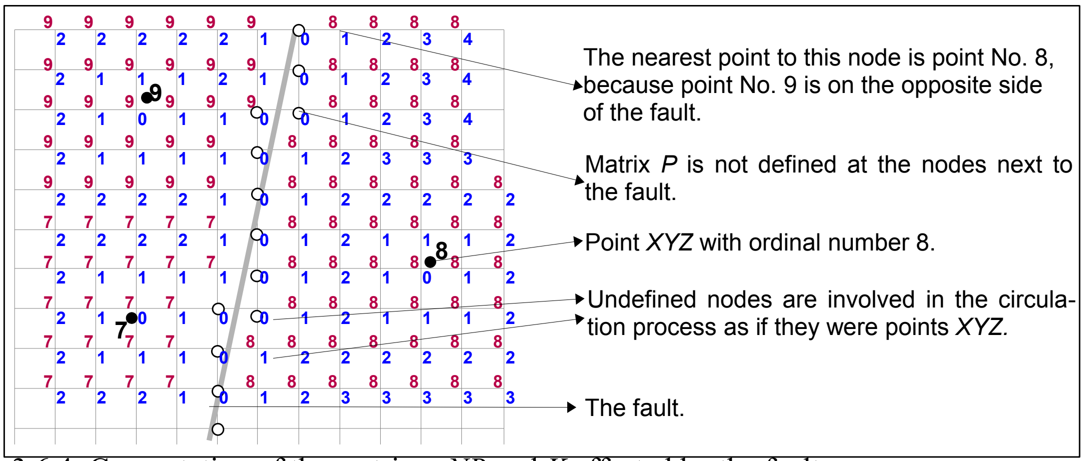

The nearest point to this node is point No. 8, because point No. 9 is on the opposite side of the fault.
Matrix *P* is not defined at the nodes next to the fault.
Point *XYZ* with ordinal number 8.
Undefined nodes are involved in the circulation process as if they were points *XYZ*.
The fault.
9
8
7

Fig. 3.6.4: Computation of the matrices *NB* and *K* affected by the fault.

The above-presented rules ensure that the points involved in tensioning cannot lie on the opposite side of a fault.

During the tensioning or smoothing process, only defined elements of the matrix *P* are used for the computation of weighted average.

3.7 Limits of the actual compilation

The actual compilation contains several limits as for the maximal number of faults, boundaries and so on.

The number of *XYZ* points (including added points and the points generated from spatial polylines) was limited to 300000, but starting from version 6.50, it is limited only by available computer memory. The original limit was set with respect to an acceptable time for the filtering process (see section ***3.4.1 Implementation of filtering***). After implementation of the super-block search strategy, even millions of points can be filtered in reasonable time.

The maximal number of vertices in one spatial polyline is 10000.

The maximal number of boundary polylines is 100 and the total number of all line segments creating a boundary is 10000.

The maximal number of fault line segments is 1000.

3.8 Formats of input and output files

3.8.1 Convention for file names

The ABOS implementation uses a special convention for naming files containing input data. The file name must have the name in the form **NAME.XXs**, where the **NAME** is an arbitrary name of a project, the **XX** is a two character part of the extension indicating what kind of data is contained in the file and the **s** is a one-character suffix enabling to distinguish between related sets of map objects (for example layers). In this way the map objects are stored in ordinary ASCII files without requiring a database system. This convention is utilized namely by the *SurGe Project Manager*, as described in section ***4.1 Project manager***.

3.8.2 Points

The basic input file is an ordinary ASCII file which has a name in the form **NAME.DTs**, where **NAME** is the name of the project, **DT** is the extension indicating the type of data (points *XYZ*) and **s** is the suffix. Each row of the file has this format:

```
X  Y  Z  L
```

where real numbers **X**, **Y** and **Z** are x, y and z coordinates of the points *XYZ* and **L** is the label of the point containing max. 23 characters. Items in a row must be separated by at least one space. The file containing added points **NAME.DBs** has exactly the same format.

The basic input file is the only file, which can have comment rows starting with the character # in the first column.

3.8.3 Spatial polylines

The file containing spatial polylines must have a name in the form **NAME.LNs**. The file has this format:

N_1 M_1
X Y Z
X Y Z
.
.
N_2 M_2
X Y Z
X Y Z
.
.
N_p M_p
X Y Z
X Y Z
.
.

In the first row of each sequence of the spatial polyline points (vertices), there must be the number of points in the sequence ($N_1, N_2, ..., N_p$). The second and the next rows (**X Y Z**) contain x, y and z coordinates (real numbers) of polyline vertices separated by at least one space. The number of polyline vertices (N_i) is limited to 10000.

$M_1, M_2, ..., M_p$ are the numbers of internal points (see ***3.6.2 Spatial polylines***).

3.8.4 Boundary

The file containing boundary polylines has a name in the form **NAME.HR.** There is no suffix because the boundary is expected to be common for all maps in the project. The file has this format:

N_1
X Y
X Y
.
.
N_b
X Y
X Y
.
.

In the first row of each sequence of the boundary points, there must be the number of points in the sequence ($N_1, N_2, ..., N_b$). The second and the next rows (**X Y**) contain x and y coordinates (real numbers) of the boundary points separated by at least one space. The overall number of boundary points (**N_1+N_2+...+N_b**) cannot exceed 10000. The number of boundaries (**b**) is limited to 100.

3.8.5 Faults

NAME.ZL is the name of an ASCII file containing coordinates of initial and end points of the line segments, at which the created surface has to be discontinuous. Similarly as in the case of the boundary there is no suffix, because the faults are expected to be common for all maps in the project. The file has this format:

```
X1 Y1 X2 Y2
X1 Y1 X2 Y2
.
.
```

The coordinates must be separated by at least one space. The number of lines in the file cannot exceed 1000. The line segments can be connected and so they can form a polyline. They are often referred to as faults.

3.8.6 Grids

The output ASCII file containing the grid has the name **NAME.GRs**. It contains the matrix **i1**×**j1** of z-coordinates in the nodes of the grid. The format of the file is compatible with Surfer (Golden Software) grid file format:

```
DSAA
i1 j1
x1 x2
y1 y2
z1 z2
P1,1    P1,2   P1,3   .... P1,i1
P2,1    P2,2   P2,3   .... P2,i1
.       .      .
.       .      .
Pj1,1   Pj1,2  Pj1,3  .... Pj1,i1
```

In addition to an ASCII grid file, SURGEF creates a binary grid file named **NAMEf.GRs** with the following records:

```
i1 j1 x1 x2 y1 y2 z1 z2
P1,1    P1,2   P1,3   .... P1,i1
P2,1    P2,2   P2,3   .... P2,i1
.       .      .
.       .      .
Pj1,1   Pj1,2  Pj1,3  .... Pj1,i1
```

The binary grid file is approximately five times smaller than the ASCII grid file and it is used for communication between SURGEF and the graphical user interface SurGe.

3.9 Interface for user applications

The program SURGEF, which implements the interpolation / approximation method ABOS, can be used as an external program called from user application, for example:

- Using the **system** command in C language
- Using the **Shell** function in Microsoft Visual Basic
- Using the **WinExec** function, which is available in standard Windows library KERNEL32.DLL.

To run SURGEF.EXE in this way, the application must provide:

1. Input file(s) for SURGEF.EXE (at least the basic input file must exist in the working directory).

2. The application must create the ASCII file PAR.3D (parametric file for SURGEF.EXE) with items described in the following table:

Row	***Value***	***Example***	***Meaning***
1.	String	ex1	Name of basic input file
2.	Character	A	One-character suffix
3.	Y / N / C [,scale]	C,1.2	Boundary has to (Y) / does not have to (N) be used. If C is used, the boundary will be created as a convex envelope of input points. The following optional number can then be used as a scale of boundary (default value is 1.1).
4.	Y / N	N	Faults have to (Y) / do not have to (N) be used
5.	Y / N	N	Additional points have to (Y) / do not have to (N) be used
6.	Y / N	N	Polylines have to (Y) / do not have to (N) be used
7.	Y / N	Y	Basic points have to (Y) / do not have (N) to be used
8.	0-9999	500	Value of filter
9.	5-99 [, 0 / 5-5555, 0 / 5-5555]	99, 300, 200	Grid enlargement, gird dimension(s)
10.	0-4, 0 / 1	1, 1	Degree of linear tensioning, fast convergence off (0) / on (1)
11.	0-99, 0-999.99 [,0-9999]	1, 0.5, 50	Precision, smoothing, number of smoothing cycles
12.	Y / N	N	Blank (Y) / do not blank (N) grid outside the boundary
13.	Y / N	N	Create (Y) / do not create (N) NP file (see ***4.2.11.2***)
14.	Y / N	Y	Create (Y) / do not create (N) ASCII grid file
16.	[file-name-xy]	points.xy	Input file containing x and y coordinates of points as the first two items
17.	[file-name-xyz]	points.xyz	Output file containing records from file-name-xy + surface values at specified points

Values in brackets are optional. If a numeric parameter is missing, it will be estimated by SURGEF internally.

If there are two or more items in the row, they can be separated by a comma and spaces or only by spaces.

If both grid dimensions are zero, they will be estimated by SURGEF internally. If the first (dimension in x-direction) is greater than zero and the second (dimension in y-dimension) is zero or is missing, the second dimension will be estimated by SURGEF internally according to the first dimension.

If the C option is used in the third row, the new boundary created as a convex envelope will replace the old boundary, if it exists. Practical usage of the convex envelope is described in section ***5.6 Digital model of terrain***.

If a boundary has to be used, the domain rectangle *D* of the interpolation function is set according to boundary points and not according to the points *XYZ*.

3. SURGEF.EXE can be called from the application – for example by the following command in C language (do not forget the N parameter, which means "normal" interpolation; other interpolation modes are used for development purposes).

```
system("C:\SurGe\SURGEF.EXE N");
```

3.10 Running SURGEF.EXE

Even if SURGEF is intended for running from another applications (namely from GUIs), it can also be run in the console window like this:

`E:\Fprog\Surgefr\data>SURGEF N`

In the working directory there must be the parameter file PAR.3D (see the previous section) and at least a basic input file containing points *XYZ*. If there are other files containing other map objects (polylines, faults, boundary), they will be involved in interpolation depending on the information contained in the parameter file.

The dump of a typical console screen running SURGEF.EXE is shown in the following frame:

```
E:\Fprog\Surgefr\data>SURGEF N

SURGEF - v.6.30
(c) M.Dressler 1996 - 2007
Compiled with GNU Fortran G77 v. 3.4.2

READING DATA: --
Filtering:  13504 --- 13411 ---- 13156 ------- 12253 -------------- 9643
NUMBER OF INPUT POINTS WAS REDUCED FROM:   13504
                                     TO:    9643
READ FILE .GRD? (Y/N) [N]
GRID SIZE IN X-DIRECTION (MIN.  500) [ 500]: 600
GRID SIZE IN Y-DIRECTION (MIN.  348) [ 417]:
GRID SIZE ENLARGEMENT [ 62]:
NEAREST POINTS: ----------------------------------------------
FAULTS CHECKING: ---------------------------------------------------------
Dynamically allocated memory:     10.60 MB
DR=0.00174  DF=0.02371  DR/DF=0.07358
NP= 63  NQ= 150  JH= 33  GL=  1.20279877
SMOOTHING [ 99]: 100

*****************
* APPROXIMATION *     (Press Esc to stop iterations)
*****************

INTERPOLATION: OK
TENSIONING: ----------------------------
TENSIONING: ---------------------------
SMOOTHING:  -----
GRADIENT:   ----------------------------------------------------------
SMOOTHING:  ---------------------------------------------

 Average deviation =   1.48878503
 >>>>>>>>>>>>> RELATIVE PRECISION >>>>>>>>>>>>>  4.986 [%] - POINT 9316

INTERPOLATION: OK
TENSIONING: ---------------------------
SMOOTHING:  -----

 Average deviation =   0.136240065
 >>>>>>>>>>>>> RELATIVE PRECISION >>>>>>>>>>>>>  0.872 [%] - POINT 9497

Time elapsed:   4.37  /   4.00  [sec]

BINARY GRID IS BEING CREATED:
```

In the rows containing the text in red (prompts), SURGEF offers default or suggested values and expects a response from the user. The user can leave the suggested value by pressing the **Enter** key or can enter a new value (examples can be seen in blue).

If **`Y`** (yes) is answered to the prompt **`READ FILE .GRD? (Y/N) [N]`**, the existing grid file is read as the initial interpolation / approximation function. It means that the vector *DZ* is not initialised as vector *Z* in the first step of the interpolating algorithm (see ***2.2 Interpolation algorithm***), but its values are computed as $DZ_i = Z_i - f(X_i, Y_i)$. Moreover, the read grid can be smoothed first, because in this case the following additional prompt is displayed:

```
SMOOTHING OF READ GRID [   0]:
```

The default value 0 means that no smoothing will be performed.

The following two prompts

```
GRID SIZE IN X-DIRECTION (MIN.  500) [ 500]: 600
```

and

```
GRID SIZE IN Y-DIRECTION (MIN.  348) [ 417]:
```

are intended for changing the default grid size suggested by SURGEF. The grid size in the y-direction is suggested so that the difference between *Dx* and *Dy* is minimal (see ***2.2.2 Specification of the grid***). If any of the grid sizes are smaller than the minimal value, there is a high probability that the iteration process will not converge.

The **`GRID SIZE ENLARGEMENT [  62]:`** asks for the number of grid rows and columns, which are used for enlarging the interpolation function domain (see ***3.4.3 Smoothing and tensioning on grid boundary***). The value suggested by SURGEF should be left unchanged – changing this value is intended only for development purposes.

The last prompt **`SMOOTHING [  99]:`** `100` enables to change the number of smoothing cycles. The suggested value is sufficient for obtaining a smooth surface, but for example if a trend surface has to be obtained, the value may be higher (even several times).

To run SURGEF without waiting for prompt entries, i.e. automatically using suggested values, the second command line parameter **`A`** can be used:

```
E:\Fprog\Surgefr\data>SURGEF N A
```

Chapter 4
Graphical user interface

The goal of the graphical user interface design is to create an appropriate environment as a superstructure above the ABOS method implementation satisfying the following requirements:

1. management of projects
2. transformation of map objects coordinates
3. specification of interpolation parameters and running SURGEF.EXE
4. 2D and 3D display of surfaces, computation and display of isolines and display of cross-sections
5. digitisation of map objects
6. mathematical operations with surfaces
7. computation of volumes between surfaces

The first requirement is implemented in SurGe Project Manager described in the first section of this chapter.

Requirements two to six are implemented in SurGe, the main program creating the graphical user interface.

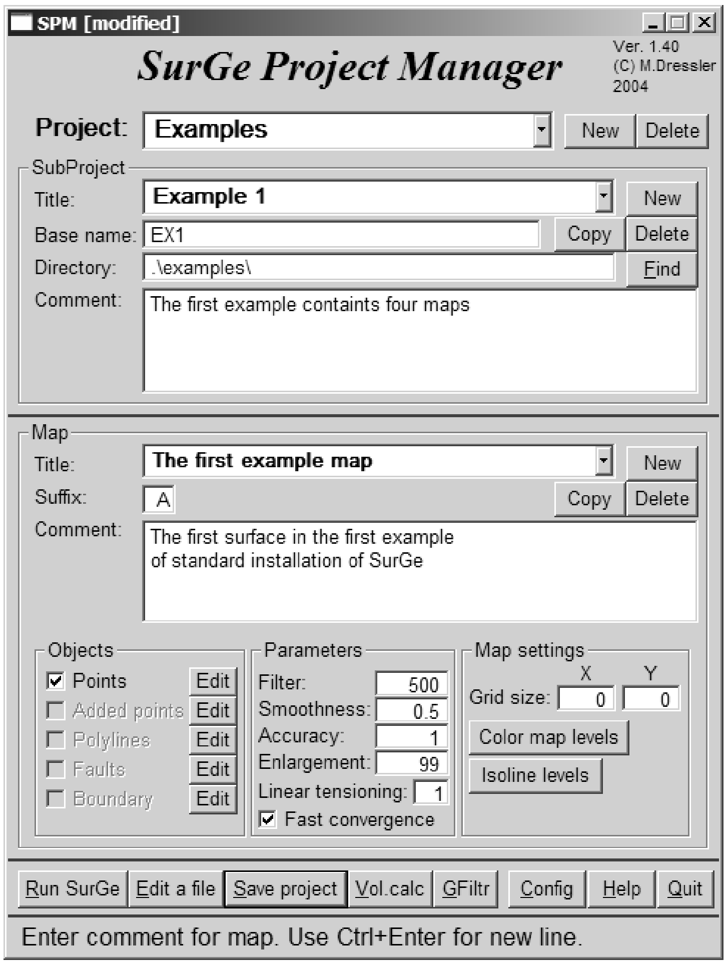
The seventh requirement is solved as a stand-alone utility VOLUME.

4.1 Project manager

SurGe Project Manager (SPM) is a simple application, which enables:

- to manage projects and maps in an easy and comfortable way (create a new project, modify or delete an existing project, add comments to the project or map and so on)
- to select map objects, which have to be included in the interpolation process
- to select interpolation parameters separately for each map in the project
- to run the SurGe graphical interface for a selected map
- to edit the data of map objects (using the stand-alone editor FMEW or using an editor selected by the user)
- to calculate volumes between two surfaces (using the stand-alone utility VOLUME)

SPM is created as a dialog-based Windows application:


SPM [modified]

SurGe Project Manager

Ver. 1.40
(C) M.Dressler
2004

Project: Examples

SubProject
Title: Example 1
Base name: EX1
Directory: .\examples\
Comment: The first example containts four maps

Map
Title: The first example map
Suffix: A
Comment: The first surface in the first example
of standard installation of SurGe

Objects
- [✓] Points
- [ ] Added points
- [ ] Polylines
- [ ] Faults
- [ ] Boundary

Parameters
Filter: 500
Smoothness: 0.5
Accuracy: 1
Enlargement: 99
Linear tensioning: 1
[✓] Fast convergence

Map settings
Grid size: X 0, Y 0

Enter comment for map. Use Ctrl+Enter for new line.

Fig. 4.1: SurGe Project Manager – the dialog-based application for managing SurGe projects.

The usage of SPM is quite intuitive and does not need a detailed description. Just a few points should be emphasized:

- The description of projects is saved in files with extension .PRO. The name of a project is the name of the corresponding file – that is why the project name must only contain allowed file name characters (for example characters * or ? are not allowed) and the name should be unique.
- The project file **EXAMPLES.PRO** is a part of the installation and contains two subproject examples (**Example 1** and **Example 2**) related to maps in the EXAMPLES directory.
- Subprojects are managed according to the subproject title. This means that the subproject title must be unique (subprojects with the same title are not allowed). The same rule holds for the map title.
- The path to the subproject directory must end with the back slash character "\" and should be absolute (for example **C:\surge\examples\**). The project file **EXAMPLES.PRO** uses the relative path (**.\examples**) in order to address files installed in the directory EXAMPLES.
- If a multiple line comment has to be entered, the shortcut key **Ctrl+Enter** must be used to start a new line. The key **Enter** has another function – see the next point.
- If the project, subproject or map has been changed, the main window bar indicates it with the text "**[modified]**". Before starting SurGe (using the button "**Run SurGe**") or before switching to an already existing window running SurGe, it is recommended to

save the project using the button "**Save Project**" or by the **Enter** key – then SurGe will read actual map parameters immediately after a new run or after switching to an already existing window running SurGe. In this way, the user can comfortably experiment with interpolation parameters.

- If the grid size in the x-direction is zero (in **Map settings**), SURGEF suggests an appropriate value. If the grid size in the x-direction is positive and the grid size in the y-direction is zero, SURGEF accepts the first value and suggests an appropriate second value. If both values are positive, SURGEF accepts them.
- The "**Edit**" button can be used for editing files containing map objects. The default editor is FMEW, but the "**Config**" button enables to enter the full path to another editor suitable for the user.
- The button "**Vol. calc**" runs the stand-alone program VOLUME for volume calculations – see section ***4.4 Calculation of volumes***.
- The last row of the SPM dialog box shows a short hint for a selected dialog item.

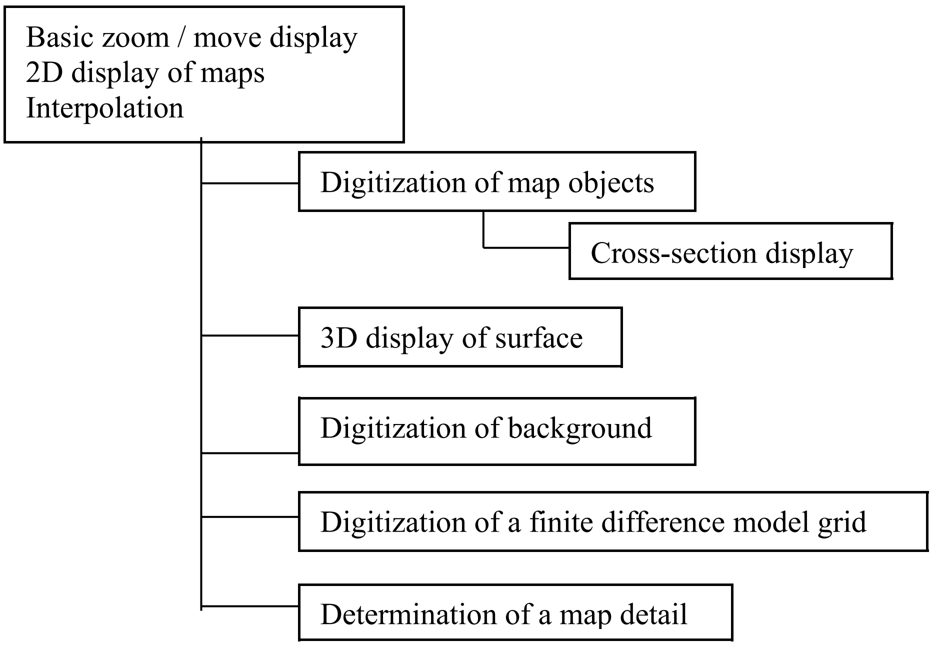
4.2 SurGe

SurGe is the main graphical program providing full interface to the ABOS implementation. It can be run from SurGe Project Manager or directly using command line statement with arguments in the form:

```
C:\MAPS>SurGe NAME s
```

where the **`NAME`** is the name of the project and **`s`** is the suffix (see paragraph ***3.8.1 Convention for file names***).

SurGe works in several levels described in the subsequent scheme.


Basic zoom / move display
2D display of maps
Interpolation
Digitization of map objects
Cross-section display
3D display of surface
Digitization of background
Digitization of a finite difference model grid
Determination of a map detail

The next paragraphs in this section describe all essential functions of SurGe.

4.2.1 Display of map objects

In the basic move / zoom display there are points *XYZ* as blue dots. If boundaries and / or faults and / or polylines exist, they are displayed too. The boundaries are displayed as thick red lines, faults as thin green lines and polylines as thin orange dotted lines. In the move / zoom mode, the display can be moved using cursor keys and zoomed by shortcut keys **PgUp** or **PgDn**. If only basic objects have to be moved / zoomed, **Ctrl** with these keys can be used too. The step of moving and zooming can be changed with shortcut keys "**1**", "**2**", "**3**", "**4**" or "**5**".

Additional displays can be performed using the items in the **Display** menu.

- Labels and z-coordinates of *XYZ* points can be displayed using the menu item **Labels** (shortcut key **N**) **Z-coordinates** (shortcut key **K**), respectively.
- The menu item **Color scale** (shortcut key **Alt+S**) displays points (and labels and / or z-coordinates, if they are displayed) in colours indicating their z-values.
- **Mesh scale** (shortcut key **Ctrl+S**) enables to display a square mesh showing distances (if the mesh has to be labelled, shortcut key **Ctrl+E** can be used). The suggested size of the mesh can be altered by the user in the presented dialog.
- **Refresh** (shortcut key **R**) is intended for restoring the basic display.
- Objects in the background (see ***4.2.8 Background***) can be displayed using the **Background** menu item (shortcut key **O**).
- Background colour can be switched between black and white using the menu item **B/W background colour** (shortcut key **Ctrl+R**).
- If there are cross-sections saved in the file **NAME.RZY** (see ***4.2.7 Cross-section display***), they can be displayed using **Saved cross-sections** (shortcut key **Ctrl+C**).

4.2.2 Transformation of coordinates

The coordinates x and y of the basic map objects (points, boundaries, faults and polylines) can be transformed. Transformation functions are contained under the main menu **Transformation**:

- The first one (**Move to beginning of coordinate system**, shortcut key **Alt+0** (zero)) moves coordinates of the basic map objects into the beginning of the coordinate system – this means that the minimal x-coordinate and the minimal y-coordinate is zero.
- The next two, **Transformation x[i]=MaxX-x[i]** (shortcut key **Alt+Z**) and **Transformation y[i]=MaxY-y[i]** (shortcut key **Alt+Y**), mirrors map objects according to the x axis or the y axis respectively.
- Coordinates x and y can be interchanged using the item **Interchange x and y coordinates** (shortcut key **Alt+Z**).
- All objects can be rotated - counter clockwise using the menu item **Rotation counterclockwise** (shortcut key **Alt+U**) or clockwise using the menu item **Rotation clockwise** (shortcut key **Ctrl+U**).
- The **Scaling and translation** menu item invokes the dialog box enabling to scale and translate all coordinates by specified constants.
- The last item **Save objects** is intended for saving all objects into corresponding files (see ***3.8 Formats of input and output files***).

4.2.3 Interpolation

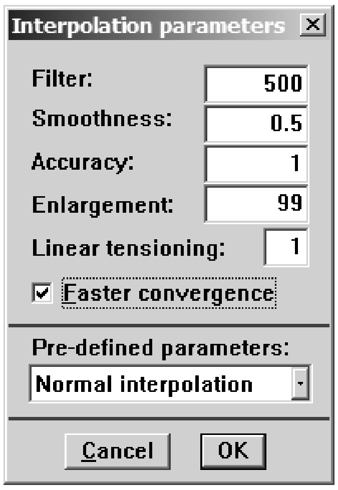
The main menu item **Interpolation** enables to specify map objects, set interpolation parameters, run the interpolation process, compute isolines and so on. The subsequent paragraphs describe these functions.

4.2.3.1 Objects for interpolation

The first item in the **Interpolation** sub-menu contains a selection of map objects, which have to be entered into the interpolation process. They are **Points**, **Added points**, **Polylines**, **Faults** and **Boundaries** (see section ***3.6 Map objects***).

4.2.3.2 Interpolation parameters

The quality of the surface generated by the program SURGEF can be changed in the dialog box invoked by **Interpolation parameters** menu item:



Filter: 500
Smoothness: 0.5
Accuracy: 1
Enlargement: 99
Linear tensioning: 1
[✓] Faster convergence
Pre-defined parameters: Normal interpolation

Fig. 4.2.3: Dialog box for the specification of interpolation parameters.

The first parameter **Filter** (the default value is 500) is intended for reducing input points if the number of points is very large and if there are points with a small horizontal distance between them. Usage of the parameter **Filter** was explained in paragraph ***2.2.1 Filtering of points XYZ***.

The parameter **Smoothness** (see paragraph ***2.2.7 Smoothing***) enables to control smoothness of a generated surface. The larger the value, the sharper interpolation is obtained. Typical values are:

0,00 - 0,30 for smooth interpolation
0,40 - 0,60 for normal interpolation (default value is 0,50)
0,70 - 1,50 for sharp interpolation.

A sharp/smooth model at local extremes can be improved by extending the smoothing parameter. Beginning from SurGe version 5.50, the smoothing parameter can have two formats:
1) number 0,00 - 9,99, which is equivalent to the above described smoothing parameter
2) number 100,00 - 999,99, where the first two digits divided by 10 determine a so called shape factor, which has an influence on the shape of the surface in the surrounding of sharp local extremes. The smallest value 1.0 means, the shape will not be changed and any greater value (1.1 - 9.9) means that the local extreme will be sharper. The remaining digits have the original meaning.

Remark: If the smoothing parameter has the first format, the shape factor has the default value 1.0.

Parameter **Accuracy** (the default value is 1) is the percentage value from the difference $z2-z1$. The role of this parameter was described in paragraph ***2.2.8 Iteration cycle***.

Enlargement is the grid size enlargement described in section ***3.10 Running SURGEF.EXE***. If it is greater than 98, the program SURGEF computes it internally.

The parameter **Linear tensioning** enables to set the degree (0-3) of linear tensioning (see ***2.2.6 Linear tensioning*** and ***3.4.2 Degrees of linear tensioning***). The default value is 1.

In most cases the number of iterations can be decreased (see ***2.2.8 Iteration cycle***) by the transformation $a \cdot P_{i,j} + b \rightarrow P_{i,j}$, where constants *a* and *b* minimize the term

$$\sum_{i=1}^{n} (a \cdot f(X_i, Y_i) + b - DZ_i)^2$$

The resulting surface is somewhat smoother, but the number of iterations is decreased by cca 30%. The check button **Faster convergence** enables this feature.

The pull-down list **Pre-defined parameters** contains the list of interpolation / approximation modes and enables to set appropriate parameters for a selected mode. The modes and pre-defined parameters are:

Mode	*Filter*	*Smoothness*	*Accuracy*	*Linear tensioning*
Trend Surface	30	0,1	90	0
Smooth approximation	200	0,2	20	0
Smooth interpolation	500	0,2	1	1
Normal interpolation	500	0,5	1	1
Sharp interpolation	500	200,7	1	1
LES interpolation	1000	-0,5	1	1
Digital model of terrain	1000	200,7	1	3

Important note: The *Trend surface* and *Smooth approximation* set a special multiplier for the SMOOTHING parameter otherwise estimated by SURGEF. To deactivate this setting, the *Normal interpolation* item should be selected.

4.2.3.3 Interpolation

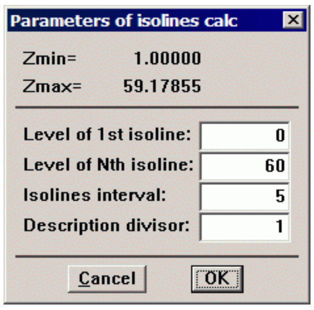
The interpolation / approximation process is started using the **Calculate grid** menu item. Firstly, the parameter file PAR.3D is created and then SURGEF is run in a new console window. The content of a typical console window running SURGEF is described in section ***3.10 Running SURGEF.EXE***.

4.2.3.4 Increasing the density of the grid

There is the possibility to double the grid (using the menu item **Double grid**) once or more times. Z-values of newly created grid nodes are computed by means of quadratic interpolation. A doubled grid provides better isolines and it can be used for the creation of an extra smooth surface. Of course, each doubling creates a file four times greater in size.

4.2.3.5 Calculation of isolines

Before displaying, isolines must first be calculated. The calculation is invoked by the menu item **Calculate isolines**. The following dialog appears:


Parameters of isolines calc

Zmin= 1.00000
Zmax= 59.17855

Level of 1st isoline:	0
Level of Nth isoline:	60
Isolines interval:	5
Description divisor:	1

The meaning of individual items in this dialog is apparent, but four points should be emphasized:

- Only the isolines having a level divisible by the divisor will be labelled.
- If a small difference between isolines is selected, the calculation can last from several seconds to minutes.
- The calculated isolines are stored in the binary file **NAME.VRs** where the **s** is a suffix (see paragraph ***3.8.1 Convention for file names***) and then they are immediately displayed.
- If the surface is later created with different interpolation / approximation parameters, the isolines should be recalculated to correspond to the actual surface.

An example of printed isolines is in figures 2.4.2b, 2.4.2c and 2.4.2d.

4.2.3.6 Blanking grid outside the boundary

The function **Blank grid outside boundary** is intended for cancelling values of the grid nodes located outside the boundary. To obtain isolines only inside the boundary, the calculation of isolines must be then performed again. Examples of this function are in sections ***5.4 Wedging out of layer*** and ***5.6 Digital model of terrain***.

4.2.3.7 Cutting off extreme values

The function **Substitute below** enables to substitute all z-values of the surface, which are less than a specified constant, by this constant. For example, negative values of the grid nodes can be substituted with zero. A similar function has the menu item **Substitute above**. An example of this function is in section ***5.1 Zero-based maps***.

4.2.3.8 Mathematical calculations with grids

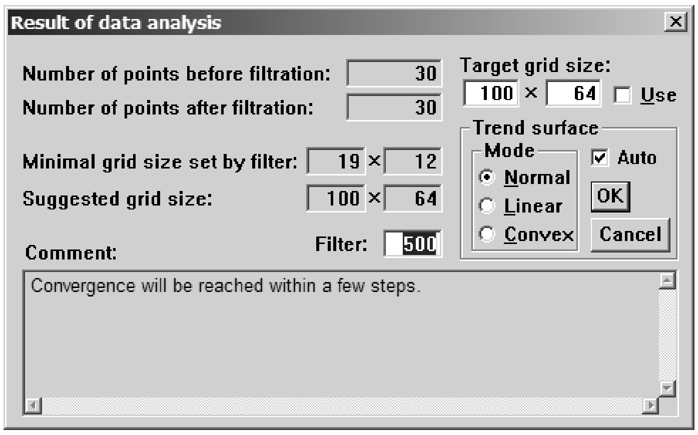
The menu item **Math calculation with grids** starts a dialog box enabling to perform some calculations with all nodes of grids. It is assumed that the first operand is the actual surface and the second one is a previously created surface defined by the suffix. If the second operand is not defined (the suffix is empty), it is assumed to be a constant (specified in the following dialog box). The result of the operation is indicated by one character with the following meaning:

Operand	***Result***
~	Negation
+	Addition
-	Subtraction
*	Multiplication
:	Division
m	Minimum
M	Maximum
a	Average
$	the first operand; if the second is greater than the first, then average
%	the second operand; if the second is greater than the first, then average
w	weighted average (the weights are specified in the following dialog box)
d	derivative computed as the size of gradient vector

Examples of these functions can be found in sections ***5.4 Wedging out of a layer*** and ***5.5 Maps of thickness and volume calculations***.

4.2.3.9 Data analysis

The **Data analysis** menu item runs only the first part of SURGEF.EXE to get essential information about filtering, grid sizes and expected maximal gradient. Then it displays the following dialog box:


Result of data analysis

Number of points before filtration: 30
Number of points after filtration: 30
Minimal grid size set by filter: 19 × 12
Suggested grid size: 100 × 64
Comment:
Filter: 500
Target grid size: 100 × 64 ☐ Use
Trend surface
Mode: (● Normal) ○ Linear ○ Convex
☑ Auto
OK
Cancel

Convergence will be reached within a few steps.

The first and second items inform about the number of points before and after the filtration process. If the grid size is smaller than the **Minimal grid size set by filter**, there is a high probability that the iteration process will not converge. The **Suggested grid size** is a grid size suggested by SURGEF.EXE. The **Comment contains** a verbal description of data analysis results and some suggestions.

The edit box **Filter** enables to change the actual setting of the filter (and, for example, to run **Data analysis** again to observe its influence). The **Target grid size** has two purposes:

1. If interpolation with a trend surface is performed, this grid size will be used without respect to the state of the **Use** check box.

2. If the check box **Use** is switched on, this grid size will be used for the next interpolation / approximation and for the next data analysis.

The items **Normal**, **Linear**, **Convex** and **Auto** in the **Trend surface** group box are intended only for interpolation with a trend surface (see the next paragraph).

4.2.3.10 Interpolation with a trend surface

Interpolation with a trend surface runs SURGEF.EXE two or three times. The first run creates a trend surface with a small grid having the following properties:

- the grid size of the small grid is between 80 and 160 or between 40 and 80 or between 20 and 40 – it depends on the selection of the **Preservation of extrapolation trend** parameter 1, 2 or 3 in the provided dialog box
- the **Target grid size** (see the previous section) is the 2^n multiple of the small grid size.

The second (third) run reads the created grid of the trend surface doubled n-times and then performs a modified interpolating algorithm as described in section ***3.10 Running SURGEF.EXE***. Using this procedure the trend surface is involved in the interpolation, meaning that the resulting surface keeps a proper trend in areas without points. It is recommended to perform data analysis before interpolation with a trend surface and to set a desired **Target grid size**.

An example of interpolation with a trend surface is in paragraph ***2.4.3 Conservation of an extrapolation trend*** and in section ***5.2 Extrapolation outside the XYZ points domain***.

4.2.4 Display of surface

2D displays of the created surface can be performed using the following items in the **Display** menu.

- Isolines can be displayed (assuming that they were calculated) using the item **Isolines** or by the shortcut key **I**.
- The surface can also be represented as a colour raster map using **Colour Map** (shortcut key **C**).
- The menu item **Shadowed relief** (shortcut key **Alt+Q**) enables to display a shadowed colour map improving the 3D feel of the display (see figure 4.2.4); the angle and intensity of the shadow is specified in the presented dialog.
- The colour of the base objects (points *XYZ*, labels, **z**-coordinates, boundaries, faults and polylines) can be switched using **Change colour** (shortcut key **Ctrl+A**) in order to achieve better visibility of these objects on the colour map.
- There are three items related to the gradient display. The first one, **Gradient in nodes**, shows gradient as short oriented line segments starting at the nodes of the grid. The second one, **Gradient in isolines**, shows similar line segments starting along the isolines (only if isolines have been calculated). In both cases the user can change (in the provided dialog box) the multiplier constant (default 100) specifying the length of the line segments and the frequency (default 2). For example, frequency=2 means, the gradient line segments will start in every second node. When the function **Gradient lines** is selected, the program enters digitisation mode. In this mode the cursor has the shape of a little cross and the cursor keys move the cursor (and not the map). The gradient lines (starting from the cursor position) can be displayed using the shortcut key **Alt+G**.

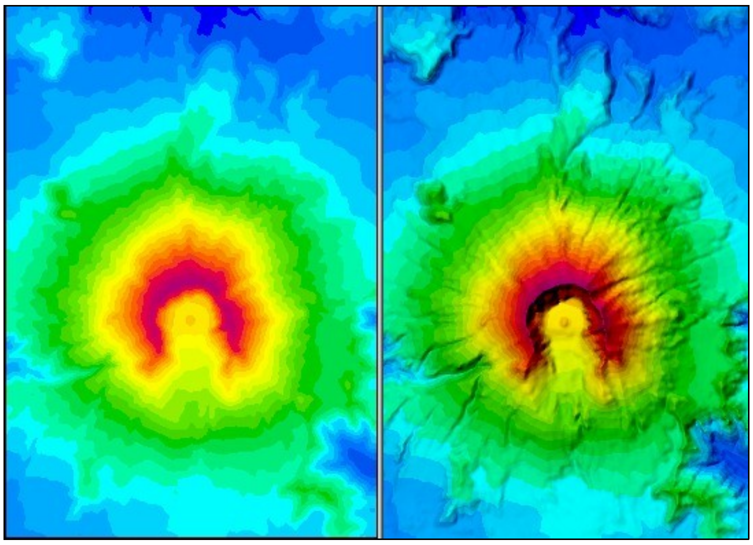


Fig. 4.2.4: Difference between colour map and shadowed colour map display.

4.2.5 3D display

The menu item **Display / 3D view** is intended for displaying and viewing the created surface in 3D from different angles and different elevations. In this case the surface is firstly stored to the file **NAMEF.GRs** and then is read again. Before reading the surface, there is a possibility of changing the step of reading node values. The default step value is 1, which means, that all grid nodes will be displayed. A value of 2 means, that every other grid node will be displayed and so on. Higher values enable to display a sparser grid.

Cursor keys can be used for moving the 3D view around the screen, shortcut keys **PgDn** and **PgUp** are intended for zooming.

In the 3D view mode there is a special menu having the following items:

- The surface can be displayed without colours (**Display wireframe surface**, shortcut key **S**), with colours (**Display colour surface**, shortcut key **C**) or as a shadowed colour surface:
    - Shortcut key **E** displays the surface with pure colours and clears all shadows.
    - Shortcut key **Alt+Q** adds shadows with angle and intensity specified in the presented dialog. It can be used several times with different angles and intensities to achieve nice lighting effects.
    - Shortcut key **W** displays the surface with the actual settings of shadows (after rotation, zoom, …).
- The horizontal angle of the view is changed by **Rotate counterclockwise** (shortcut key **A**) or **Rotate clockwise** (shortcut key **Shift+A**).
- The vertical angle can be changed by **Rotate up** (shortcut key **B**) or **Rotate down** (shortcut key **Shift+B**). The items **Increase z-scale** (shortcut key **Z**) and **Decrease z-scale** (shortcut key **Shift+Z**) are intended for increasing and decreasing the superelevation, respectively.
- The item **Display / hide labels** (shortcut key **K**) switches on/off the display of point labels.
- Background colour can be changed using **B/W background** colour (shortcut key **Ctrl+R**).

Step of angles, moving and zoom can be changed with shortcut keys "**1**", "**2**" and "**3**". The horizontal and vertical angles of view can also be changed by clicking and holding the left mouse button and moving the mouse.

4.2.6 Digitisation

Digitisation is intended for manipulating basic map objects. After selecting one of the items **Points**, **Boundaries**, **Faults** or **Polylines** in the menu **Digitisation**, the program enters digitisation mode with a special menu. In this mode the cursor has the shape of a little cross and cursor keys change the position of the cursor (and not the position of the map as in the basic move / zoom display mode). The step of the cursor movement can be changed with shortcut keys "**1**", "**2**", "**3**", "**4**" or "**5**". Of course, the location of the cursor can be changed using the mouse too. While the cursor is being moved, the main window bar shows the coordinates of the cursor.

4.2.6.1 Points

The menu **Points** enables to add or delete points *XYZ* or change their z-coordinate. A new point is specified by the cross cursor position when the left mouse button or the shortcut key **B** is pressed. The point next to the cursor position is deleted using the right mouse button or the shortcut key **Ctrl+B**. The shortcut key **Shift+B** is intended for setting a new z-coordinate of the point, which is next to the cursor position.

4.2.6.2 Boundaries

The menu item **Boundaries** is intended for creating and correcting boundaries. Boundaries are handled as a horizontal polyline.

The shortcut key **Ctrl+H** ends the definition of one boundary and starts a new one. A new point of the boundary polyline (including the first one) is defined by the position of the cursor when the shortcut key **H** is pressed. The last entered point can be deleted using the shortcut key **D**. The shortcut key **U** is intended for closing the boundary (for creating a boundary as a closed curve). Any point of the boundary can be marked using the shortcut key **Shift+H** and then moved to a new position by the shortcut key **M**. The last item under the menu **Boundaries** creates a boundary as a convex envelope (convex hull) of points *XYZ*. The number entered in the following dialog box enables to scale the convex envelope. Using this feature the size of the interpolation function domain can be changed and grid values outside of the convex envelope can be removed – see section ***5.6 Digital model of terrain***.

4.2.6.3 Polylines

Spatial polylines can be digitised using the menu **Polylines**.

The shortcut key **Ctrl+L** ends the creation of one polyline and starts another. A new point is defined using the shortcut key **L**, the last entered point can be deleted using the shortcut key **D**. As in the case of the boundary, the polyline can be closed using the shortcut key **U**.

When defining a new point, the corresponding z-coordinate must be specified in the provided dialog box. If the z-coordinates of the polyline are to be the same, there is the possibility to predefine a constant z-coordinate using the shortcut key **F**. This function can be cancelled using the shortcut key **Ctrl+F**.

Any point of the polyline can be marked using the shortcut key **Shift+L** and then moved to a new position by the shortcut key **M**.

If a polyline is created, it is necessary to set its number of internal points. In fact, SURGEF does not work with polylines directly – it works only with the points, which are evenly distributed along the polyline. To specify the number of these points, move the cross cursor near the polyline and press the shortcut key **P**. Then in the provided dialog box enter the number of internal points (typical values are 50 - 200).

4.2.6.4 Faults

The menu item **Faults** enables to edit sequences of line segments (in horizontal plain), at which the resulting surface has to be discontinuous. Each fault is defined by a pair of points. A new point of the fault is specified by the cursor position when the shortcut key **Z** is pressed. The fault can be deleted using the shortcut key **D** – the one whose centre is closest to the cursor position. The position of the fault end point can be changed by the shortcut key **Shift+Z**.

4.2.6.5 Isolines

In the digitisation mode there is also the possibility to modify isolines. The first function, **Mark isoline** (shortcut key **X**), serves for selecting an isoline. After selection, the isoline is represented by a sequence of white points. If the display is (due to operations) damaged, it can be restored using the menu item **Redraw modified isoline** (shortcut key **A**).

The selected isoline can be smoothed (**Smooth isoline**, shortcut key **S**) as a whole, or partially (**Smooth between points**, shortcut key **V**) between the points selected using the menu items **Mark first point** (shortcut key **Alt+1**) and **Mark second point** (shortcut key **Alt+2**).

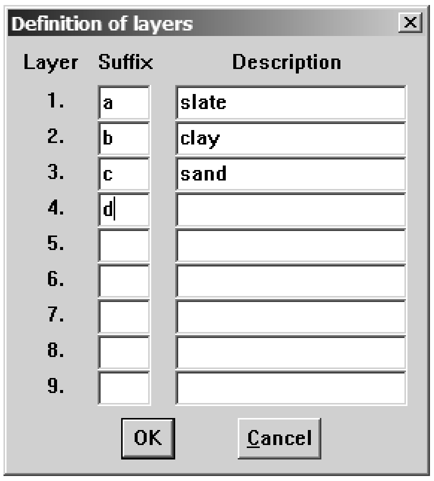
There is also the possibility to mark a single isoline point (**Mark point**, shortcut key **Shift+I**) and to move it (**Move point**, shortcut key **M**). A certain number (the number can be changed using **Change number n**, shortcut key **Ctrl+Q**) of isoline points can be moved by **Move n points** (shortcut key **Q**). The modified isoline can be saved to the file **NAME.VRz** using the menu item **Save modified isoline** (shortcut key **Ctrl+I**). **Write marked isoline** (shortcut key **W**) enables to store the selected isoline as the ASCII file **IZ.$ $$**.

4.2.6.6 Cross-sections

The menu **Cross-section** enables to specify a polyline in the plane (x,y), which defines the cross-section through the created surface. The first or next point of the cross-section can be specified using the menu item **Point of cross-section line** (shortcut key **G**). Points are displayed in red. A polyline connecting the red points is displayed using the menu item **Display cross-section line** (shortcut key **E**). All specified cross-section points can be deleted by **Delete specified cross-section** (shortcut key **Ctrl+G**) and then a new cross-section can be defined. In the cross-section mode (see below), there is a possibility to save the position of the cross-section and name it with a single letter. The function **Select saved cross-section** (shortcut key **Shift+G**) enables to select one or more saved cross-sections. When the cross-section is defined, the shortcut key **Enter** is used for creating the cross-section through the surface(s) and entering cross-section mode.

4.2.7 Cross-section display

Cross-section mode is invoked by the menu item **Cross-section** / **Display cross-section** (shortcut key **Enter**) in the digitisation mode. The following dialog box appears:


Definition of layers

Layer	Suffix	Description
1.	a	slate
2.	b	clay
3.	c	sand
4.	d	
5.		
6.		
7.		
8.		
9.		

The suffix convention (see paragraph ***3.8.1 Convention for file names***) enables to create a cross-section through several surfaces (layers). In the presented dialog box, the suffix of the surface and a short description can be specified. The cross-section is constructed through all specified surfaces and displayed in a 2D plot (see for example figure 2.3).

There is a special menu in the cross-section mode. The title of the plot and description of axes can be modified using menu items **Graph title** (shortcut key **N**), **Description of x axis** (shortcut key **X**) and **Description of y axis** (shortcut key **Y**). The range of the y axis can be changed by **Change range of y axis** (shortcut key **Ctrl+Y**).

The last menu item, **Save cross-section** (shortcut key **U**) enables to save the position of the cross-section polyline to the file **NAME.RZY** for later use in the digitisation mode (see the previous paragraph). The name of the cross-section must be specified as a single character A-I.

4.2.8 Background

In some cases it is useful to display texts and some characteristic terrain lines and / or objects to improve orientation in the map. For this purpose, there is a special digitisation level named **Background** under the menu **Digitisation**. Objects of background are handled as polylines and the following functions (contained in the special menu) for creating, changing or deleting objects from background can be used:

Start new object (Ctrl+B)
- ends construction of the current object and starts construction of a new one.

New point of object (B)
- a new point of a polyline is specified at the cursor position.

Delete last point (D)
- the last specified point is deleted.

Close polygon (U)
- the next point will be at the same position as the first point of the polyline.

Mark point (Shift+B)
- change the colour of the nearest point of the polyline to the cursor to white.

Move point (M)
- move marked point to a new position.

New text (T)
- specify text string, font size, font thickness, font colour and text orientation.

Change text (Shift T)
- change text string, font size, font thickness, font colour and text orientation.

Three functions are intended for manipulation with whole objects:

Select object/text (S)
- select / unselect the object next to the cursor. The selected object is displayed in purple.

Move selected object/text (Ctrl+M)
- move whole object; the moving vector is given by the marked point and the cursor position.

Delete selected object/text (Ctrl+D)
- delete selected object.

All background objects can be saved to the file **NAME.BG** with the last menu item **Save background** (shortcut key **W**).

4.2.9 Finite difference model grid

The function **Model grid** in the menu **Digitisation** enables to create or to modify an irregular rectangular grid for a mathematical finite difference model, for example MODFLOW. The program switches into a special digitisation mode with its own menu. In the menu there is a list of shortcut keys, which can be used for creating an irregular rectangular grid:

Add column (shorcut key **X**)
- a column line is added at the cursor position

Add row (shorcut key **Y**)
- a row line is added at the cursor position

Delete column (shortcut key **Ctrl X**)
- the column near the cursor position is deleted

Delete row (shortcut key **Ctrl Y**)

- the row near the cursor position is deleted

Mouse adds / deletes columns (activated using shortcut key **Shift X**)
- clicking the left mouse button adds a column and the right mouse button deletes the nearest column

Mouse adds / deletes rows (activated using shortcut key **Shift Y**)
- clicking the left mouse button adds a row and the right mouse button deletes the nearest row

Delete whole grid (shortcut key Ctrl Z**)**
- the whole grid is deleted and a new one can be created

Display background (shortcut key **O**)
- background objects (see the previous paragraph) are displayed

Save modelled grid (shortcut key **W**)
- the model grid is stored in the ASCII file **NAME.XY** (see ***4.2.11.2 Finite difference model grid***)

Automatic grid generation (shortcut key **A**)
- a grid is generated automatically in dependence on two parameters entered by the user – the minimal block size and block size increment. The algorithm for automatic grid generation tries to ensure, that all points XYZ are as close to the grid block centre as possible and the size of grids uniformly increases in areas without points. A simple example of such a grid is in figure 4.2.9.

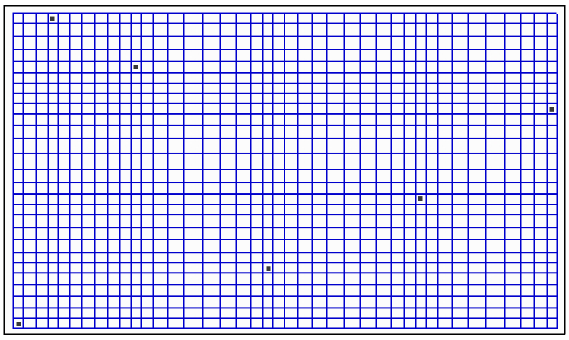


Fig. 4.2.9: Automatically created finite element model grid.

In addition to coordinates of grid lines the file **NAME.XY** also contains the list of points with grid coordinates (for example the uppermost point in figure 4.2.9 has grid coordinates 3 1 and the rightmost point has grid coordinates 40 9), which are important for the location of points (such as wells) in the mathematical model.

4.2.10 Output

The items under the **Output** menu are intended for the output of grid files in various formats extending the SurGe usability and compatibility with some GIS and gridding / mapping systems. Support for additional systems are solved as a set of stand-alone conversion utilities (see section ***4.3 Supported map formats***).
The following list contains a short description of individual menu items.

Grid as ASCII file

Z-values of the surface are stored in ASCII format in the file **NAME.GRs**. This format is compatible with the grid file format used by Surfer (Golden Software) – see section ***3.8.6 Grids***.

Remark: A grid compatible with Surfer (stored in ASCII format) can be read using the menu item **File / Read grid from ASCII file**.

Grid as GRASS file

Z-values of the surface are stored in ASCII format in the file **NAMEs.TXT**. This file, compatible with GRASS GIS system, can be imported as a raster file into LandSerf, a free application for the visualization and analysis of surfaces.

Grid as ArcGIS file

Z-values of the surface are stored in ASCII format in the file **NAMEs.GRD**. This file format is compatible with the popular ArcGIS system and it is supported by most gridding / mapping systems.

Z-values at points

This function reads x and y coordinates from the specified input ASCII file, computes corresponding z values at the surface and writes the result (x, y and z values) into a specified output file. The input file must contain x and y coordinates in the first two items of each row. The rest of the row is copied into an output file.

The format of the input file rows must be:

```
X  Y  [any-text]
```

The format of the output file rows is:

```
X  Y  Z  [any-text]
```

Such a type of output provides a very important universal tool for transferring surface z values into any set of points located in the interpolation / approximation function domain. For example, if the x and y coordinates in the input file are triangle vertexes of an unstructured grid (for example the grid of a finite element model), then this tool provides conversion between structured and unstructured grids.

Grid as DAT file

The grid values are stored in the format of the basic data file **NAME.GDs** (see paragraph ***3.8.2 Points***). The file containing grid values in this format is also referred to a generic ASCII grid file and it is used in many GIS systems such as Global Mapper.

Isolines as ASCII file

The isolines are stored in the ASCII format file **NAMEa.VRs**.

NPR file

Surface values are interpolated into block centres of the model grid and stored in the ASCII file **NAME.NPs**. The model grid **NAME.XY** (see ***4.2.9 Finite difference model grid***) must exist.

4.2.11 SurGe input / output files

In addition to files used by SURGEF (see section ***3.8 Formats of input and output files***), SurGe uses some other files for input or output. These files comply to the convention rules for file names described in paragraph ***3.8.1 Convention for file names*** and their format is described in the subsequent paragraphs.

4.2.11.1 Isolines

Isolines are stored in a binary file named **NAME.VRs**. Whenever the isolines are calculated, this file is created as a new one and the old one (if exists) is replaced. The file consists of the following records:

$Z\ N\ X_1\ Y_1\ X_2\ Y_2,\ \dots,\ X_N\ Y_N$

where Z is the isoline level, N is the number of isoline points and $X_1\ Y_1\ X_2\ Y_2,\ \dots,\ X_N\ Y_N$ are the coordinates of points creating the isoline.

Isolines can also be stored as an ASCII file **NAMEa.VRs** (using the menu item **Output / Isolines as ASCII file**) having the same format as the basic input file **NAME.DTs**.

X_1 Y_1 Z
X_2 Y_2 Z
.
.
X_N Y_N Z
.
.

4.2.11.2 Finite difference model grid

The finite difference model grid is saved in the ASCII file **NAME.XY** with the following format:

NX NY
X_1 X_2 … X_{NX}
Y_1 Y_2 … Y_{NY}
DX_1 DX_2 … DX_{NX-1}
DY_1 DY_2 … DY_{NY-1}

IX_1 JY_1 LB_1
IX_2 JY_2 LB_2
.
.
IX_M JY_M LB_M

where **NX** and **NY** are numbers of grid columns and rows respectively,

X_i i=1,…,NX	are x-coordinates of the model grid lines,
Y_i i=1,…,NY	are y-coordinates of the model grid lines,
DX_i i=1,…,NX-1	are grid block sizes in the x direction,
DY_i i=1,…,NY-1	are grid block sizes in the y direction,
IX_i JY_i	are model grid coordinates of a point having a label LB_i.

If the file **NAME.XY** exists, any map created within the project name **NAME** (specified by a suffix **s**) can be used for the creation of the ASCII file **NAME.NPs**, which contains surface values at centres of the finite difference model grid and which is saved as a matrix of real numbers containing **NX-1** columns and **NY-1** rows.

4.2.11.3 Colour map and isoline levels

To change the default setting for the colours in colour maps or for the levels of isolines, the user can specify the following files:

NAME.CLs is an input ASCII file containing z-levels for a raster colour map. If this file does not exist, default levels and colours are used. The format of this file is:

Z_1 $[C_1]$ $[C_0]$
Z_2 $[C_2]$
Z_3 $[C_3]$
.
.
Z_n $[C_n]$

where $Z_1,...,Z_n$ ($0<n<27$) are z-levels specified by the user and $C_0,C_1,...,C_n$ ($0<=C_i<61$) are optional colour numbers (colour C_i will be used between levels Z_i and Z_{i+1}). If the colour number C_0 (the default value is 1 – dark blue) is specified in the first row, it will be used below the level Z_1. If there is a row without colour specification, the default colours will be selected. The following scale shows colours and their numbers used in SurGe.

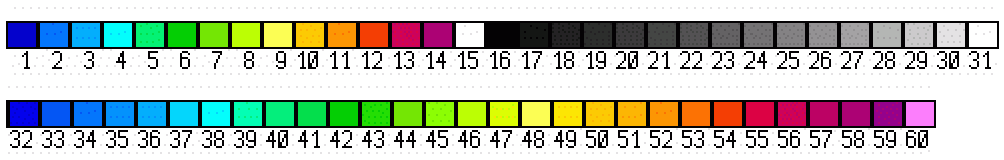


Fig. 4.2.11.3 Colour scale used in SurGe

NAME.CL is an input ASCII file with the same format and the same meaning as the file **NAME.CLs**. If this file exists in the working directory, it is used for all maps with the name **NAME**. But if there is also the file **NAME.CLs** in the working directory, it has precedence before the file **NAME.CL** for the map with the suffix **s**.

NAME.LVs is an input ASCII file containing user-defined z-levels for isolines. The format of this file is:

```
Z₁
Z₂
Z₃
.
.
Zₙ
```

where $Z_1,...,Z_n$ (`1<=n<=700`) are z-levels specified by the user. If the file exists, the levels in the file are used for the computation of isolines and the dialog for the specification of isoline levels is not displayed.

NAME.LV is an input ASCII file with the same format and the same meaning as the file **NAME.LVs**. If this file exists in the working directory, it is used for all maps with the project name **NAME**. But if there is also the file **NAME.LVs** in the working directory, it has precedence before the file **NAME.LV** for the map with the suffix **s**.

4.2.11.4 Cross-sections position

As mentioned in paragraph ***4.2.7 Cross-section display***, the position of the cross-section may be saved for later use with the same surface or with related surfaces having the same domain and differing only by suffix. The cross-section position is saved in the ASCII file **NAME.RZY** containing one or more (but at most nine) lines with the format:

$L \quad n \quad X_1 \; Y_1 \quad X_2 \; Y_2 \; \dots \; X_n \; Y_n$

where `L` is a character (A-I) designating the cross-section, `n` is the number of cross-section points and $X_i \; Y_i$ are x and y coordinates of cross-section points.

4.3 Supported map formats

4.3.1 Review of supported map formats

SurGe supports several map formats used by other mapping / gridding systems like GRASS or ArcGIS. Some of them are directly supported by SurGe, the others are supported by means of conversion command line utilities described in the next paragraph. The review of supported mapping / gridding systems and the type of their support follows.

Surfer (Golden Software) grid format
- input: menu item **File / Read grid from ASCII file**
- output: menu item **Output / Grid as ASCII file**.

7.5-minute DEM (USGS Digital Elevation Model) grid format
- input: command line utility DEMGRD.EXE for converting a DEM file to a SurGe ASCII grid file and ASCII data file (see the next section ***Conversion command line utilities***).

GRASS grid format
- input: command line utility GRSGRD.EXE for converting of a GRASS ASCII grid file to a SurGe ASCII grid file and ASCII data file (see the next section ***4.3.2 Conversion command line utilities***).
- output: menu item **Output / Grid as GRASS file**. Can be used for example as a grid input in LandSerf, a free software for visualization and analysis of surfaces (see [S4]).

ArcGIS ESRI grid format
- input: command line utility ARCGRD.EXE for converting of a ArcGIS ASCII grid file to a SurGe ASCII grid file and ASCII data file (see the next section ***Conversion command line utilities***).
- output: menu item **Output / Grid as ArcGIS file**. Can be used for example as a grid input in LandSerf, a free software for visualization and analysis of surfaces (see [S4]).

Generic ASCII grid file
- output: menu item **Output / Grid as DAT file**. Can be used for example as a grid input in DLGV32 (Global Mapper, see [S5]).

ESRI Shapefile format
- input: command line utility SHPDAT.EXE for converting an ESRI Shapefile format to SurGe data objects (see the next section).

4.3.2 Conversion command line utilities

The purpose of this package is to provide command line utilities for conversion between commonly used map formats and map objects used in the SurGe software. Up to now there are four utilities:

DEMGRD.EXE	Conversion from 7.5-minute DEM file to ASCII GRD file and ASCII basic input file.
GRSGRD.EXE	Conversion from GRASS grid file to ASCII GRD file and ASCII basic input data file.
ARCGRD.EXE	Conversion from ArcGIS grid file to ASCII GRD file and ASCII basic input data file.
SHPDAT.EXE	Conversion from ESRI Shapefile format to SurGe data objects.

If requested by users, the package will extend to other map file formats (of course, if there is a complete description of the map file format).

4.3.2.1 DEMGRD

Command line syntax: **`DEMGRD name_of_DEM_file suffix`**

Command line example:

```
C:\DEMFILES>DEMGRD MSH.DEM a
Enter resolution (minimum is 1): 2
```

`MSH.DEM`	is the name of DEM file
`a`	is the suffix

This command reads the DEM file **MSH.DEM** and creates files **MSH.GRa** (grid in ASCII format) and **MSH.DTa** (basic input data).

A 7.5-minute DEM file contains a grid file, where the size of each grid block is 30x30 meters. The number of grid nodes can be greater than 300000 (the maximum number of input points for SurGe) – that is why DEMGRD asks for "resolution". In our example the resolution is 2, which means, each second node in the x and y direction is written in the MSH.DTa file. If, for example, the resolution is 3, then every third node is written and so on.

The file **MSH.DTa** can be used as a basic input file for SurGe. The grid file **MSH.GRa** can be imported into SurGe using the menu item "**File** / **Read grid from ASCII file**".

4.3.2.2 GRSGRD

Command line syntax: **`GRSGRD name_of_GRASS_file suffix`**
Command line example:

```
C:\DEMFILES>GRSGRD SFACE.TXT a
Enter resolution (minimum is 1): 2
```

`SFACE.TXT`	is the name of GRASS ASCII grid file
`a`	is the suffix

This command reads the GRASS grid file **SFACE.TXT** and creates files **SFACE.GRa** (grid in ASCII format) and **SFACE.DTa** (basic input data).

The number of grid nodes can be greater than 300000 (the maximum number of input points for SurGe) – that is why GRSGRD asks for "resolution". In our example the resolution is 2, which means, each second node in the x and y direction is written in the **SFACE.DTa** file. If, for example, the resolution is 3, then every third node is written and so on.

The file **SFACE.DTa** can be used as a basic input file for SurGe. The grid file **SFACE.-GRa** can be imported into SurGe using the menu item "**File** / **Read grid from ASCII file**".

4.3.2.3 ARCGRD

Command line syntax: **`ARCGRD name_of_ArcGIS_file suffix`**
Command line example:

```
C:\DEMFILES>ARCGRD SFACE.GRD a
Enter resolution (minimum is 1): 2
```

`SFACE.GRD`	is the name of ArcGIS ASCII grid file
`a`	is the suffix

This command reads the ArcGIS grid file SFACE.GRD and creates files **SFACE.GRa** (grid in ASCII format) and **SFACE.DTa** (basic input data).

The number of grid nodes can be greater than 300000 (the maximum number of input points for SurGe) – that is why ARCGRD asks for "resolution". In our example the resolution is 2, which means, each second node in the x and y direction is written in the **SFACE.DTa** file. If, for example, the resolution is 3, then every third node is written and so on.

The file **SFACE.DTa** can be used as a basic input file for SurGe. The grid file **SFACE.-GRa** can be imported into SurGe using the menu item "**File** / **Read grid from ASCII file**".

4.3.2.4 SHPDAT

Command line syntax: **`SHPDAT name_of_Shapefile name_of_SurGe_file [a]`**
Command line example:

```
C:\SHAPEFILES>SHPDAT SHXYZ.SHP SHXYZ.DTa
```

`SHXYZ.SHP`	is the name of Shapefile
`SHXYZ.DTa`	is the name of SurGe basic input file.

This command reads X,Y and Z coordinates contained in the binary file **SHXYZ.SHP** and writes them in the ASCII file **SHXYZ.DTa**, which can be used as a basic input file for SurGe.

SHPDAT can convert not only points, but also boundaries, faults or polylines – it depends on the type of shape in the Shapefile (see [S6]). The following types can be converted:

TYPE OF SHAPEFILE	CONTENT	SURGE OBJECT
PointZ (type 11)	X, Y and Z coordinates of points	points XYZ (DTs) or added points (DBs)
Polyline (type 3) or Polygon (type 5)	X and Y coordinates of polyline(s)	boundaries (HR) or faults (ZL)
PolylineZ (type 13) or PolygonZ (type 15)	X, Y and Z coordinates of polyline(s)	spatial polylines (LNs)

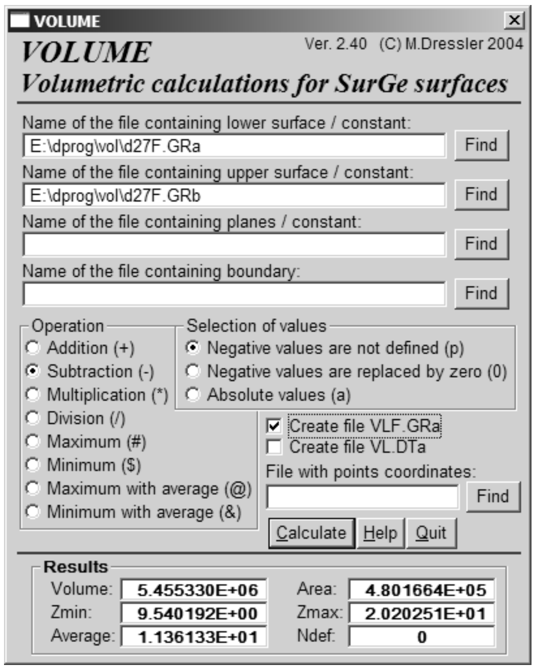
If, for example, a boundary is stored in two or more Shapefiles, parameter **`a`** can be used to create a single SurGe boundary file:

```
C:\SHAPEFILES>SHPDAT SHBOUND1.SHP SHBOUND.HR
C:\SHAPEFILES>SHPDAT SHBOUND2.SHP SHBOUND.HR a
```

SHPDAT distinguishes between a boundary and faults according to used extension in the output file name (HR or ZL) and creates appropriate data format.

4.4 Calculation of volumes

VOLUME is a dialog-based programme, which enables to calculate the volume between two surfaces created by SurGe. It can be run from Surge Project Manager (see ***4.1 Project manager***) and looks like this:



VOLUME

VOLUME Ver. 2.40 (C) M.Dressler 2004

Volumetric calculations for SurGe surfaces

Name of the file containing lower surface / constant: E:\dprog\vol\d27F.GRa

Name of the file containing upper surface / constant: E:\dprog\vol\d27F.GRb

Name of the file containing planes / constant:

Name of the file containing boundary:

Operation: Addition (+) [✓ Subtraction (-)] Multiplication (*) Division (/) Maximum (#) Minimum ($) Maximum with average (@) Minimum with average (&)

Selection of values: [✓ Negative values are not defined (p)] Negative values are replaced by zero (0) Absolute values (a)

[✓ Create file VLF.GRa]

Create file VL.DTa

File with points coordinates:

Results

Volume:	5.455330E+06	Area:	4.801664E+05
Zmin:	9.540192E+00	Zmax:	2.020251E+01
Average:	1.136133E+01	Ndef:	0

The usage of VOLUME is easy, but some points should be emphasized:

- Both upper and lower surfaces cannot be specified as constants. At least one surface must be specified as a grid file.
- If both surfaces are specified as grid files, the grids must have the same sizes and the same domains.
- The volume can be limited by planes defined in a separate file. The name of file is arbitrary, but it must have the following format:

  ```
  B₁
  X₁ Y₁ Z₁
  X₂ Y₂ Z₂
  X₃ Y₃ Z₃
  B₂
  X₁ Y₁ Z₁
  X₂ Y₂ Z₂
  X₃ Y₃ Z₃
  .
  .
  ```

  where

  X_1 Y_1 Z_1, X_2 Y_2 Z_2 and X_3 Y_3 Z_3 are coordinates of three points defining the plane,

  B_i**=1**, if the volume has to be calculated above the plane,

  B_i**=0**, if the volume has to be calculated below the plane.
- If a file containing the boundary is specified, grid nodes outside the boundary are not involved into volume computation.
- The grid obtained as a result of volume computation can be saved as a file. The name of the file is **VLf.GRa**. There is also a possibility to specify a file containing x and y coordinates. Then VOLUME will create the file **VL.DTa** with the same x and y coordinates and z values corresponding to the grid.

An example of volume calculation is in section ***5.5 Maps of thickness and volume calculations***.

Chapter 5

Solving special tasks

In the next sections there are examples of interpolation problems, which need special procedures to be solved. Most of the procedures are directly coded in the graphical user interface SurGe but some procedures are solved using stand-alone utilities.

5.1 Zero-based maps

Certain types of maps (for example maps of pollutant concentration in some area, maps of precipitation, maps of rock porosity or permeability and so on) have a common feature – their z-values cannot be negative. Let us name these maps as zero-based maps.

If a smooth interpolation is used for such types of maps, there is a real "danger" that the resulting function will be negative in some regions. As discussed in paragraph ***2.4.1 Smoothness of interpolation and oscillations***, undesired oscillations and improper extremes cannot be avoided in such cases. For this reason, the graphical user interface SurGe offers a possibility (using the menu item **Substitute below**) to substitute all z-values of the surface, which are less than a specified constant (for example zero) by this constant. Similarly, the menu item **Substitute above** enables to "cut off" values of the surface exceeding the specified constant.

As an example of a zero-based map we will use the data set CONC.DTa containing sulphate concentration measured in a soil layer. An interesting comparison of results obtained using different interpolation methods provides an evaluation of how maximal and minimal z-values of points *XYZ* were exceeded by the generated surface. As in all preceding examples, the Kriging method was used with the linear model and zero nugget effect, the Radial basis function method with the multiquadric basis functions and zero smoothing parameter and the Minimum curvature method were used with a tension of 0,1.

These are summarized in the following table.

interpolation method	***exceeding the minimal value***	***exceeding the maximal value***
Kriging	-18,8455	-4,4299
Radial basis functions	-18,8455	-4,4299
Minimum curvature	-46,4989	+0,7771
ABOS, q=0,5	-24,7731	+1,9323
ABOS, q=3,0	-16,6539	+1,8871

The difference between the surfaces created using the Kriging and Radial basis function methods is less than 1,0E-7 – that is why the results in the table are the same for these methods.

The next figure contains maps obtained using the ABOS and Kriging method. In both cases negative values were substituted with zero values. According to the opinion of many SurGe users, "pits" and "circular" contours in the surface generated by the Kriging method are undesirable.

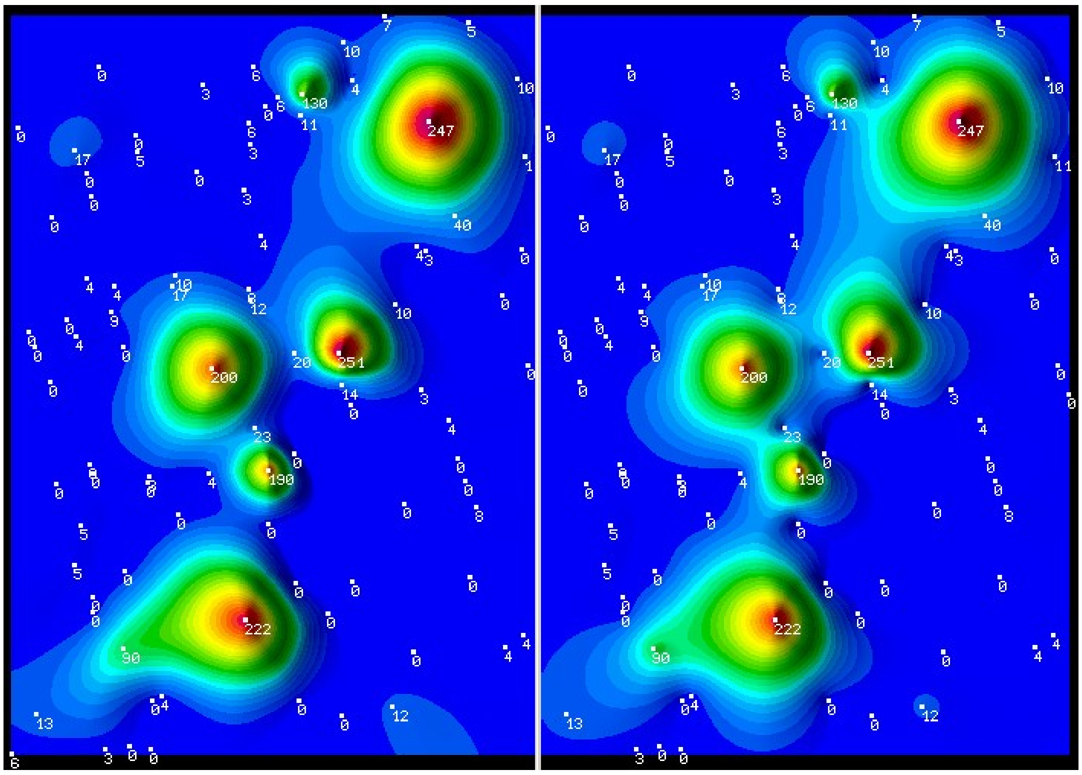


Fig. 5.1: Map of sulphate concentration created using the ABOS and Kriging method.

5.2 Extrapolation outside the XYZ points domain

The extrapolation properties of the ABOS method was examined in paragraph ***2.4.3 Conservation of an extrapolation trend***. Let us note that we only examined the domain determined by points *XYZ*. In the next example (see figure 5.2a), aerodynamic resistance data measured at a small part of a racing car body was interpreted to obtain results outside the domain determined by points *XYZ*.

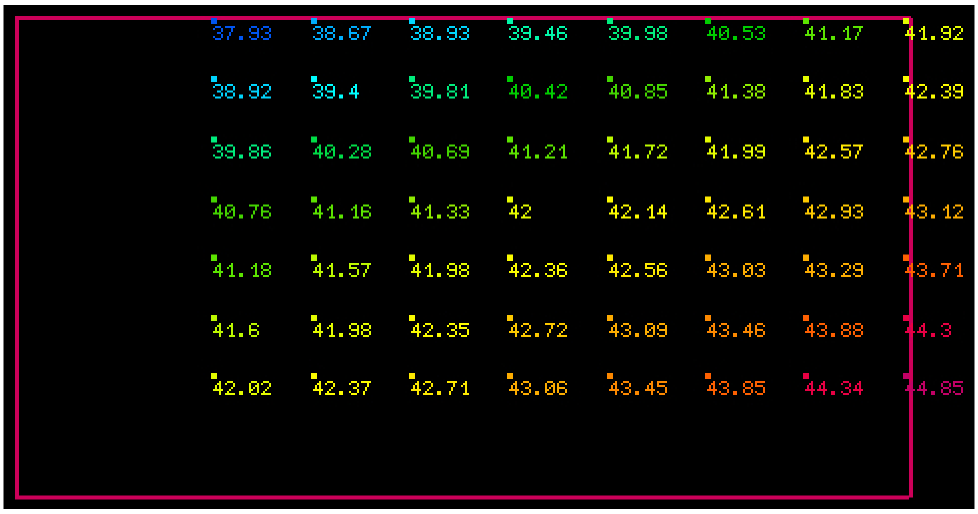


Fig. 5.2a: Aerodynamic resistance data measured at a small part of a racing car body.

As the picture indicates, the desired domain of the interpolation function set by the boundary (red rectangle) exceeds the domain defined by the points *XYZ*. The boundary was set so that the aerodynamic resistance could be estimated aside the data on the left and bottom.

The interpreter tested several interpolation methods available in the Surfer software but without satisfactory result.

As explained in paragraph ***4.2.3.10 Interpolation with a trend surface***, SurGe implements a special procedure enabling to conserve the extrapolation trend. This procedure was implemented to the examined data with the result represented in figure 5.2.b. For comparison, figures 5.2.c and 5.2.d contains results from the Kriging and Minimum curvature methods, which were not accepted as satisfactory.

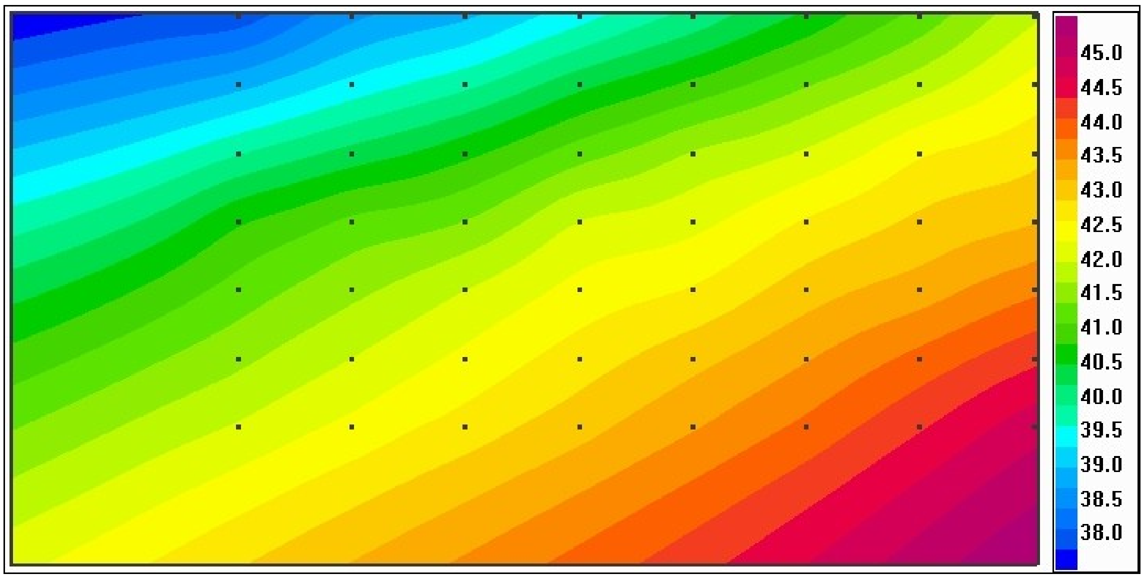


Fig. 5.2.b: Aerodynamic resistance data interpolated by SurGe.

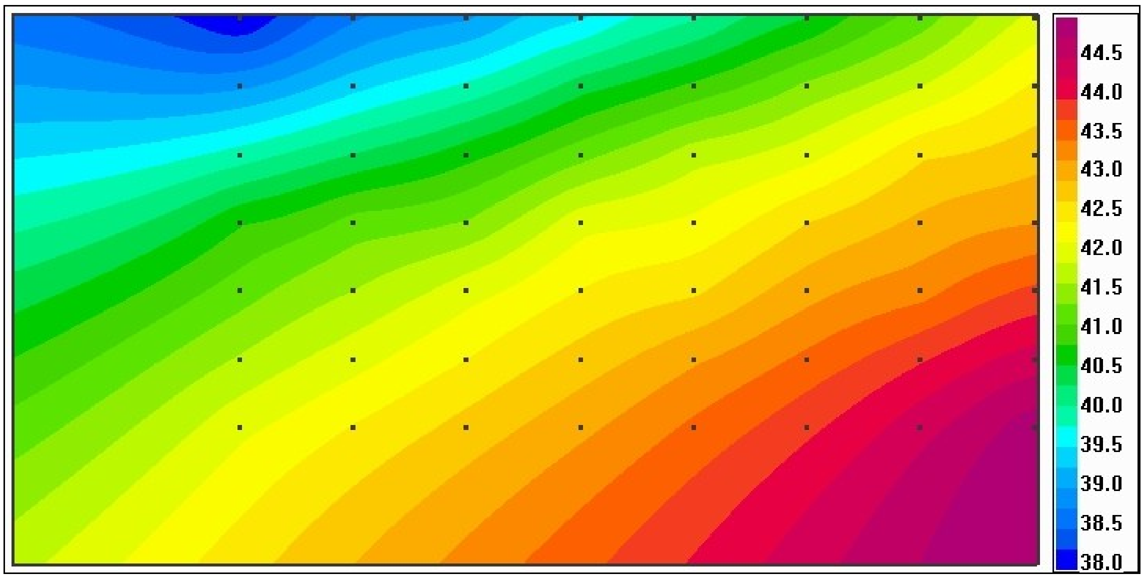


Fig. 5.2.c: Aerodynamic resistance data interpolated by the Kriging method.

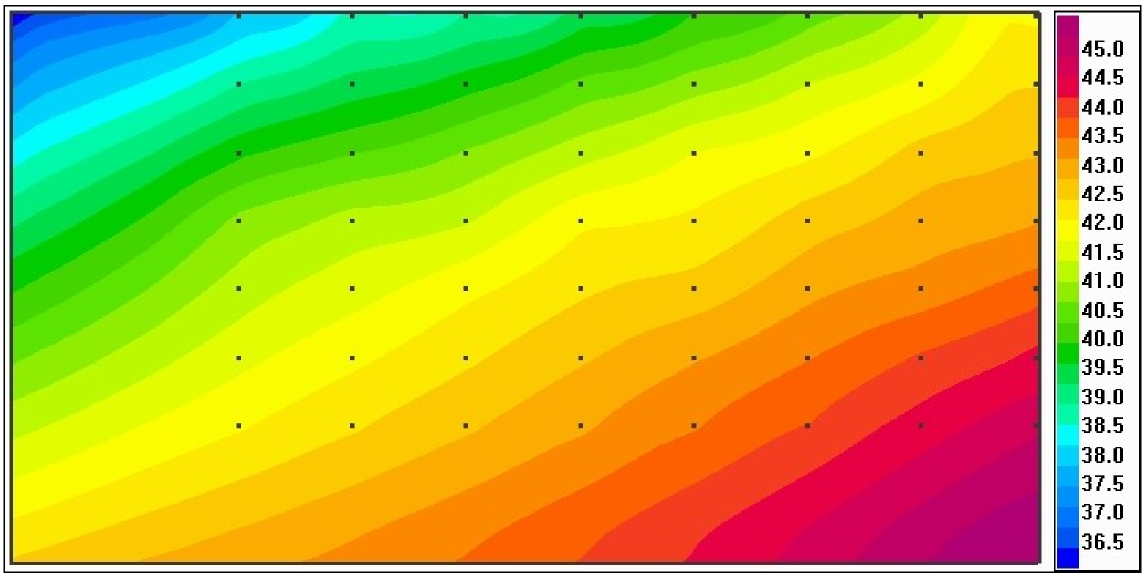


Fig. 5.2.d: Aerodynamic resistance data interpolated by the Minimum curvature method.

5.3 Seismic measurement

The processing of seismic data is one of the most significant tasks for geologists if they have to create a geological map of some underground rock structure. The typical results of seismic measurement are times at which reflected sound waves return from a certain boundary between different types of rock. These times are, of course, measured from some datum level and they are usually recorded using a dense mesh covering the area of interest with a typical number of nodes between 10000 and 100000.

Because the homogeneity of covering rocks cannot be assumed and the speed of sound waves in covering rocks is unknown or known only approximately, the measurement must be supported by precise depth values usually measured at exploration wells. In general, the interpreter has to work with two sets of data – the first one is the dense mesh of reflection times and the second one is the measurement of rock structure depth at wells. Let us note that the number of wells is usually small in comparison to the number of points at which the reflection time is measured.

To demonstrate the interpretation of seismic measurement we will use the data set SEISM1.DTc containing reflection times – the corresponding grid is shown in the next figure.

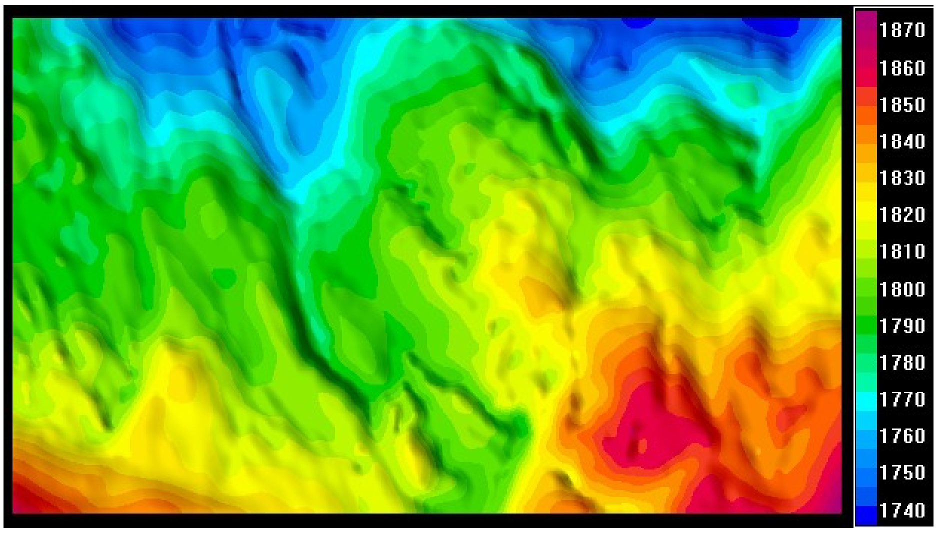


Fig. 5.3.a: Grid of reflection times created from the SEISM1.DTc data set.

The file SEISM1.DTc contains reflection times in the range from 1736 to 1875 seconds at 25853 points. It is obvious that the lower the value of reflection time, the higher the position of the rock structure boundary.

The depth of the rock structure was also measured at 14 wells and the results were stored in the file containing the second data set SEISM1.DTa – see the next list of the file content.

```
1225.976  2339.511  -2246  W-01
 837.871  2270.595  -2250  W-02
 428.004  2118.255  -2271  W-03
 859.634  1878.863  -2272  W-04
 181.357  1519.776  -2292  W-05
1940.525  2187.171  -2277  W-06
2738.498  2292.358  -2255  W-07
2981.517  2375.783  -2238  W-08
3344.232   935.805  -2338  W-09
2121.882   812.481  -2317  W-10
 493.292   500.547  -2309  W-11
1519.776  1733.777  -2267  W-12
3663.421  2143.645  -2264  W-13
2999.652  2172.662  -2252  W-14
```

As follows from the list, the range of depths is -2338 to -2238 meters. As a rule, the depth of a geological structure has a negative value measured from some datum plane (for example sea-level).

The standard procedure of seismic data processing is the following:

1. The map of reflection times is created.
2. For all wells the sound speed is calculated from known structure depth and reflection time at the well.
3. The velocities known at well positions are used for the creation of the so-called velocity map.
4. The velocity map is multiplied by the map of reflection times and thus the map of depths is obtained.

The SURGEF offers another solution: to create a depth map directly from the structure depths at wells using the reflection time grid as an external grid. This means that the grid containing the map of reflection times SEISM1f.GRc is copied (renamed) into the file SEISM1f.GRa and this grid is read at the beginning of the interpolation process. It may seem to be strange especially if we realize that the reflection times are positive values while the structure depths are negative. Let us have a closer look at the procedures performed by SURGEF.

Firstly, SURGEF is run for the data set SEISM1.DTa and it is instructed to read the external grid SEISM1f.GRa (which is a copy of SEISM1f.GRc and represents the map of reflection times) by answering **Y** to the prompt:

```
READ FILE .GRD? (Y/N) [N] Y
```

Then SURGEF changes this grid using the linear transformation $a \cdot P_{i,j} + b \rightarrow P_{i,j}$, where the constants *a* and *b* minimize the term $\sum_{i=1}^{n} (a \cdot f(X_i, Y_i) + b - DZ_i)^2$, as described in section ***2.2.8 Iteration cycle***.

In this case the constant *a* was computed as the negative number -1.160139 and together with *b* (-217.38) changed the grid so that the sum of squared differences between the new surface and structure depths at wells is minimal. As expected there were some differences between the new surface and the structural depths at wells because of heterogeneity of covering rocks; however these differences were used in the next iteration cycle (cycles) as in the normal interpolation process.

As a result, the new surface passes through negative z-coordinates of structural depths while conserving the morphology corresponding to the reflection times.

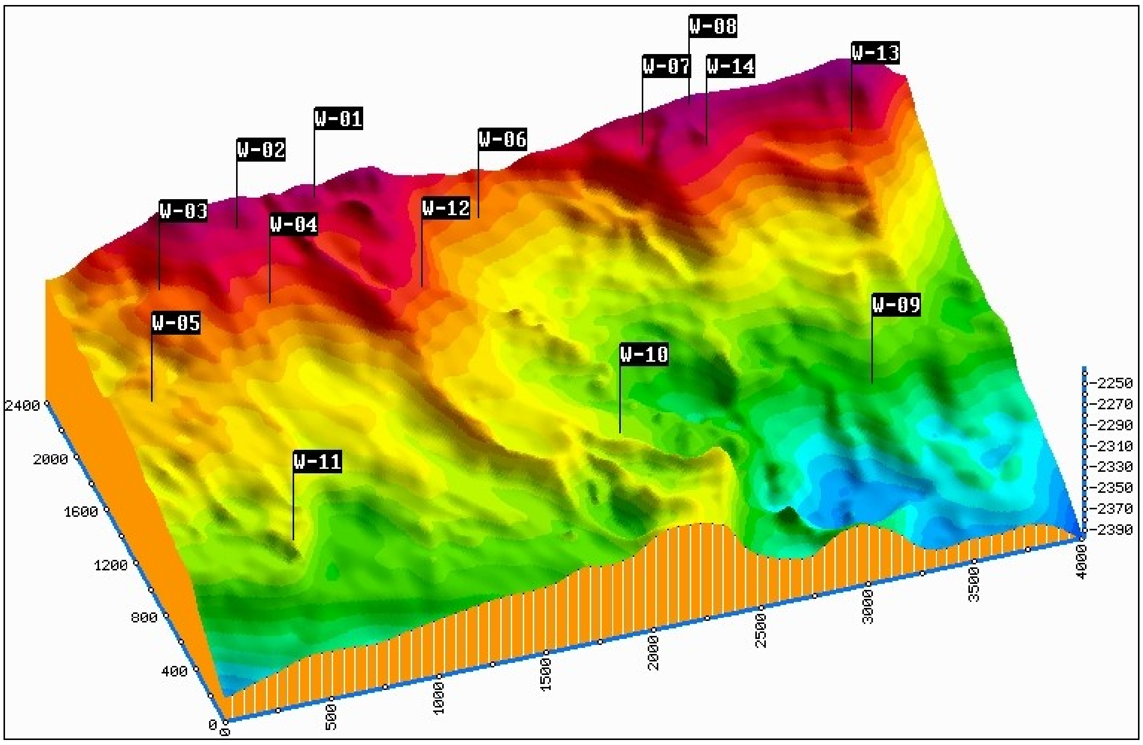


Fig. 5.3.b: Surface created directly from the seismic reflection times.

5.4 Wedging out of a layer

Construction of layer geometry is one of the basic tasks in reservoir engineering. The boundaries between individual layers are constructed as surfaces passing through structural depths (z-coordinates) measured in wells. As explained in section ***5.3 Seismic measurement***, layer construction may be combined with seismic measurement, if it is available.

If there is a small layer thickness indicated in some wells, no algorithm for smooth interpolation can ensure that the bottom layer boundary will not exceed the top layer boundary. Figures 5.4a and 5.4b illustrates such a situation – in figure 5.4a there is a top layer boundary (contained in the file GRES.DTa) and bottom layer boundary (contained in the file GRES.DTb) of a gas reservoir structure including the position of the cross-section A-A', where the bottom layer boundary exceeds the top one (see figure 5.4b). Such a phenomenon suggests that a so-called wedging out of layer (which is common in geology) should be interpreted.

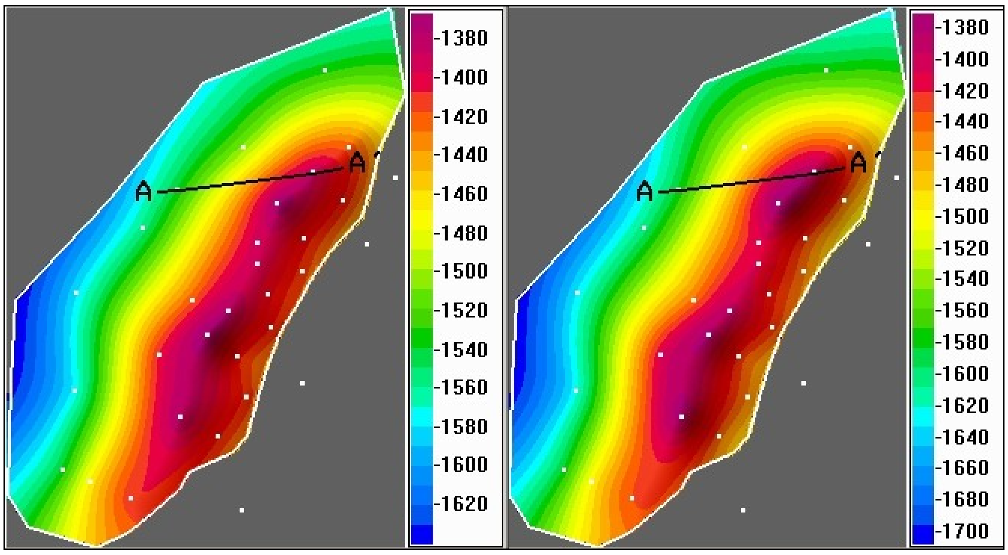


Fig. 5.4a: Maps of the top and bottom layer boundary.

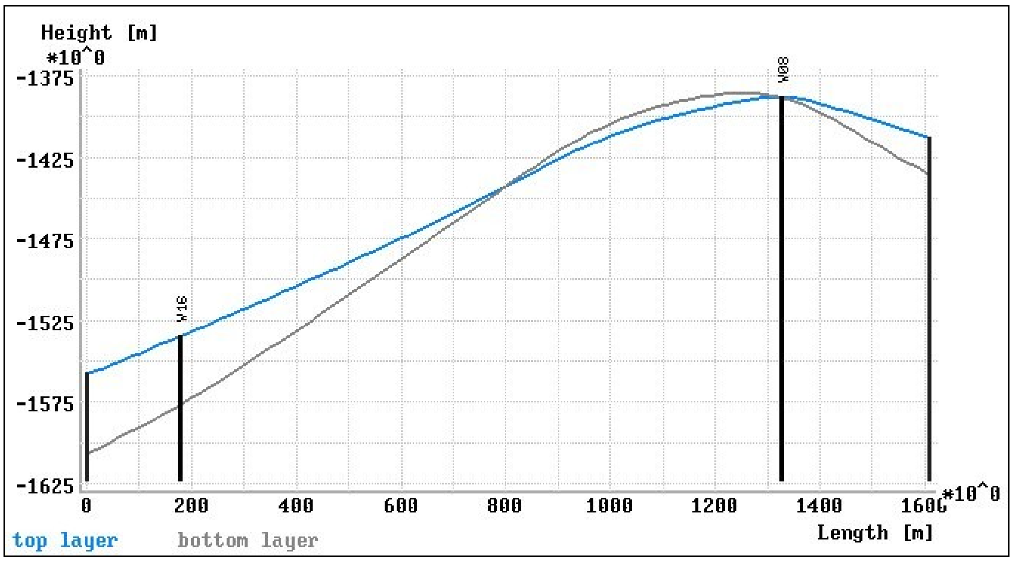


Fig. 5.4b: Cross-section A-A' through the top and bottom layer boundary.

This problem can be effectively solved in the SurGe graphical interface using mathematical operations offered in the dialog **Math calculation with grids** (see ***4.2.3.8 Mathematical calculations with grids***), where selected binary mathematical operation is performed for all z-values at nodes of grids representing the two surfaces.

There are two mathematical operations represented by characters $ and %, which can be utilized for solving the wedging out of layer problem. In the case of the first operation ($) the resulting value is the z-value of the first surface, but if the z-value of the second surface is greater, the resulting value is the average. In the case of the second operation (%) the resulting value is the z-value of the second surface, but if the z-value of the second surface is greater than the z-value of the first surface, the resulting value is the average.

Both mentioned operations were applied for the presented surfaces and the resulting new surfaces were stored with new suffixes 1 and 2. Figure 5.4c contains the cross-section A-A' through the new surfaces indicating that the wedging out of layer problem was properly solved.

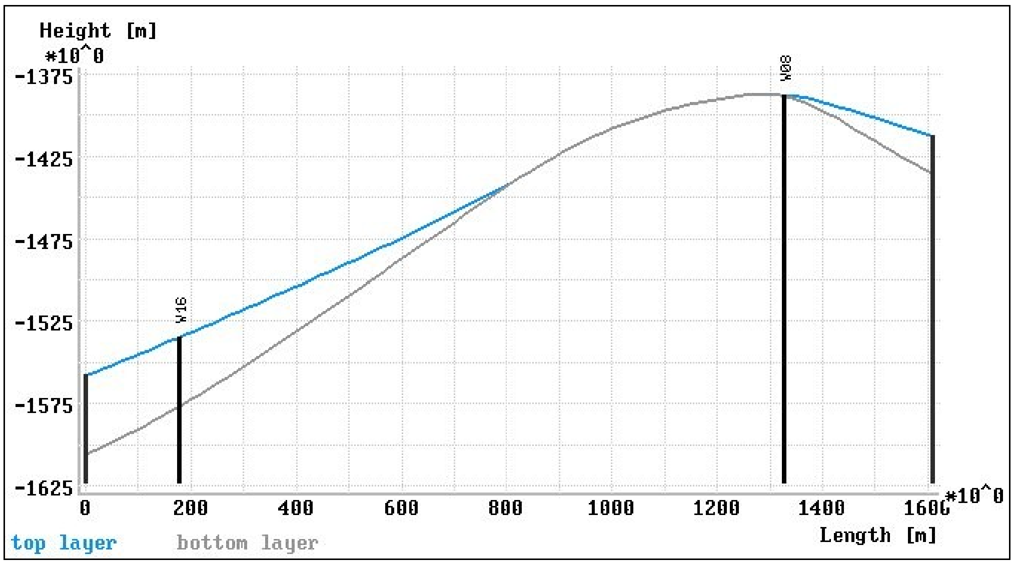


Fig. 5.4c: Cross-section A-A' through the top and bottom layer boundary after solving the wedging out of layer problem.

5.5 Maps of thickness and volume calculations

To demonstrate this feature an example concerning the estimate of new snow volume in an avalanche field after an avalanche event is presented. Two data sets were available for solving this problem – AVALAN.DT0 containing the measurement of snow surface before the avalanche event and AVALAN.DT1 containing the measurement of snow surface after the avalanche event.

In figure 5.5a there are two maps of snow surfaces created from the above-mentioned files corresponding to the situation before and after the avalanche event; the white line is an assumed boundary of the avalanche field.

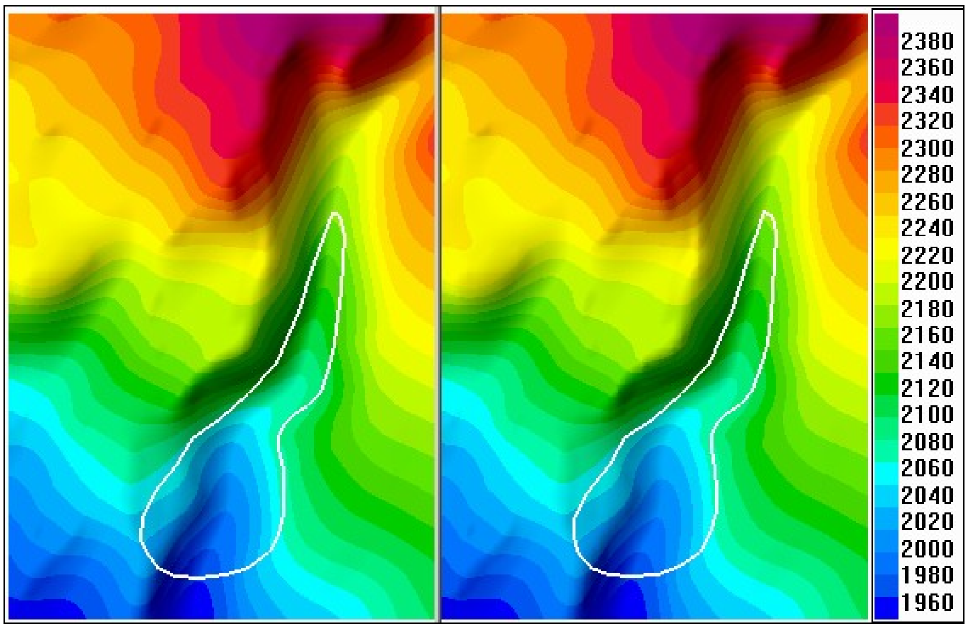


Fig. 5.5a: Snow surface before (on the left) and after (on the right) the avalanche event.

There are no visible differences between both surfaces, but as soon as the first surface is subtracted from the second (using the menu item **Interpolation** / **Math calculation with grids**), the map of the new snow thickness is obtained (see figure 5.5b).

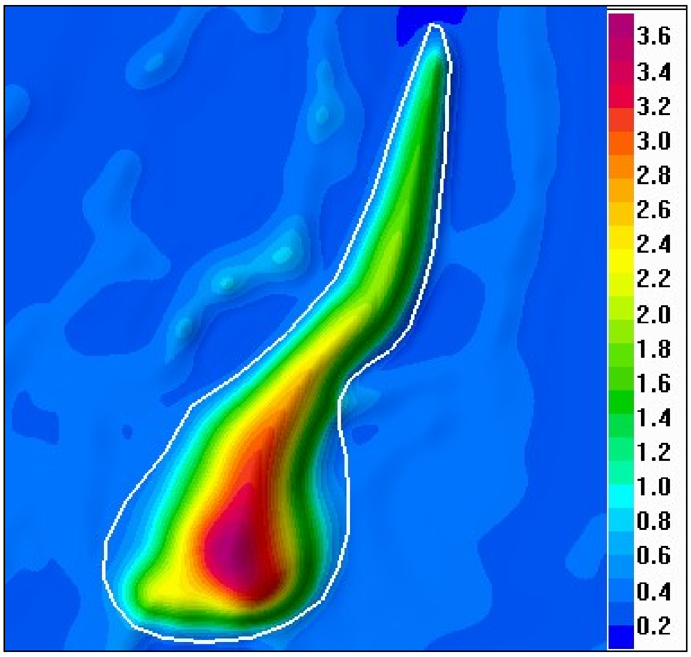
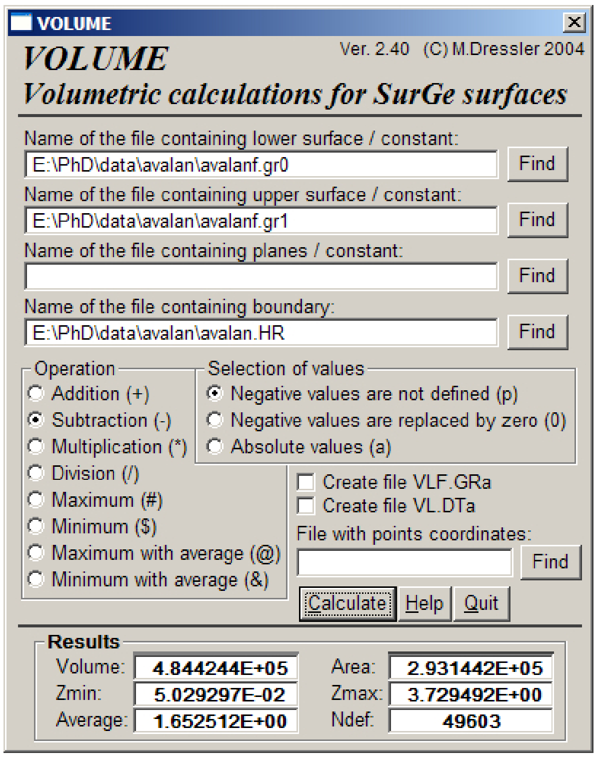


Fig. 5.5b: Thickness of the new snow layer.

It is obvious from figure 5.5b that snow also increased outside the assumed boundary of the avalanche field – it was probably caused by the additional snow precipitation, creation of snow drifts and so on.

To calculate the volume of new snow, the VOLUME utility (see ***4.4 Calculation of volumes***) can be used:



VOLUME Ver. 2.40 (C) M.Dressler 2004

Volumetric calculations for SurGe surfaces

Name of the file containing lower surface / constant: E:\PhD\data\avalan\avalanf.gr0

Name of the file containing upper surface / constant: E:\PhD\data\avalan\avalanf.gr1

Name of the file containing planes / constant:

Name of the file containing boundary: E:\PhD\data\avalan\avalan.HR

Operation: Addition (+) [✓ Subtraction (-)] Multiplication (*) Division (/) Maximum (#) Minimum ($) Maximum with average (@) Minimum with average (&)

Selection of values: [✓ Negative values are not defined (p)] Negative values are replaced by zero (0) Absolute values (a)

☐ Create file VLF.GRa
☐ Create file VL.DTa

File with points coordinates:

Results

Volume:	4.844244E+05	Area:	2.931442E+05
Zmin:	5.029297E-02	Zmax:	3.729492E+00
Average:	1.652512E+00	Ndef:	49603

Fig. 5.5c: Calculation of new snow volume using the VOLUME utility.

From figure 5.5c the following results are obtained:
The volume of the new snow layer in the avalanche field is 484424 m^3, the horizontal area of the avalanche field determined by the boundary is 293144 m^2 and the maximal thickness of the new snow layer is 3.73 m.

5.6 Digital model of terrain

The digital model of terrain is a common term for computer processing of geodetic measurement.

If the input data represents measurements from some terrain, it is usually suitable to use only linear tensioning with a small number for smoothing which is close to the Triangulation with linear interpolation method – compare digital models of a stone quarry (see [S8]) in figures 5.6a and 5.6b.

As a rule, characteristic points of terrain are measured – this means that the person performing such a measurement surveys only points where the slope of terrain changes (tops, edges, valleys and so on). For interpretation of such data, the triangulation with linear interpolation method is usually used, so the ABOS method is applicable as well.

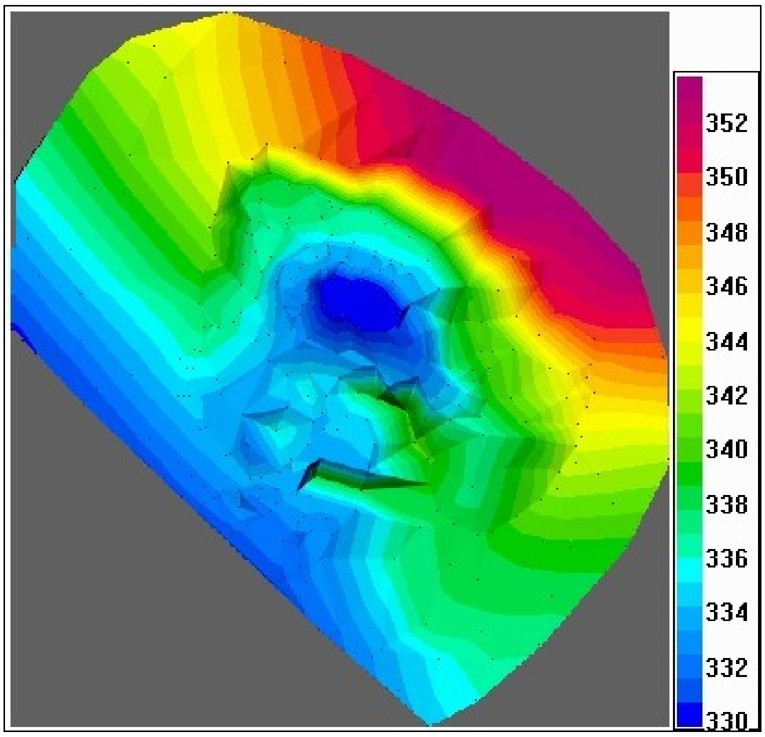


Fig. 5.6a: Digital model of the stone quarry created using the Triangulation with linear interpolation method.

As pointed out in paragraph ***1.2.1 Triangulation with linear interpolation***, the domain of the triangulation method is restricted to the convex envelope of the points *XYZ*. To restrict the domain of the function constructed using the ABOS method by the same way, a boundary as a convex envelope was created (see ***4.2.6.2 Boundaries***) and nodes outside the boundary were set as undefined using the SurGe menu item **Interpolation** / **Blank grid outside boundary**.

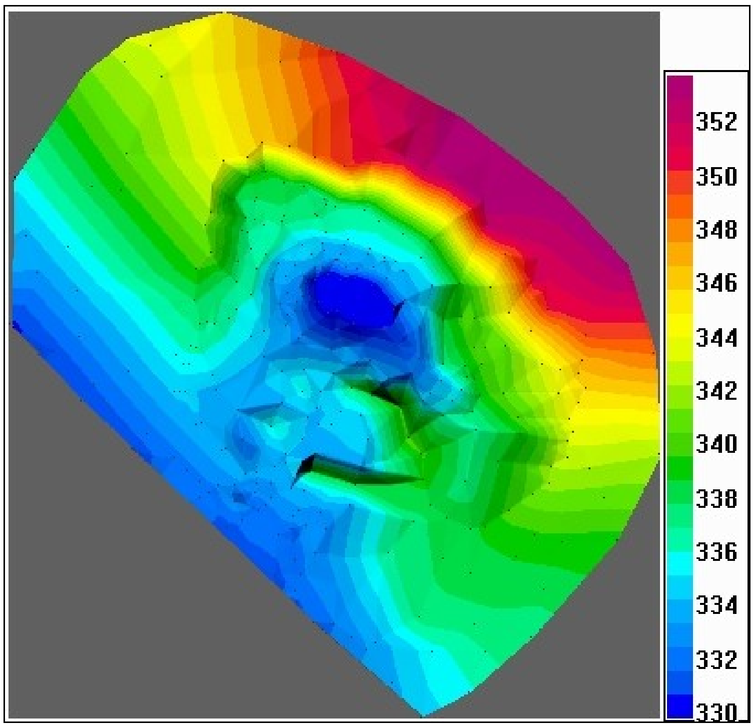


Fig. 5.6b: Digital model of the stone quarry created using the ABOS method.

5.7 Digital Elevation Model

Digital Elevation Model (DEM) is a standard (see [S9]) for the ASCII format of files containing digital geological and geographical data produced by the United States Geological Survey (USGS). It is supported by most GIS (geographic information system) applications such as Global Mapper, ARCGIS, GRASS, Geomatics, MapInfo, Intergraph, ERDAS and so on.

The data is stored as arrays of regularly spaced elevation values referenced horizontally either to a Universal Transverse Mercator (UTM) projection or to a geographic coordinate system. The grid cells are spaced at regular intervals along south to north profiles that are ordered from west to east.

In fact the DEM file is a grid file, because the elevation data (z-coordinates) are specified at nodes of a square grid (where the size of a grid square is 30 meters), but the format of this file is different from the format of a SurGe grid file.

To display DEM files using SurGe, there is a stand-alone console utility DEMGRD (see ***4.3.2 Conversion command line utilities***) included in the SurGe software enabling to convert DEM files into ASCII Surfer grid files. Moreover, the DEMGRD utility creates the basic data file, because SurGe cannot read a grid file without the basic data file. Then the user of SurGe has two possibilities – to import the grid file using the menu item **File** / **Read grid from ASCII file** or to create a new grid from the basic data file.

As an example of a DEM file displayed by the SurGe software there is an elevation model of mount Shasta in Northern California.

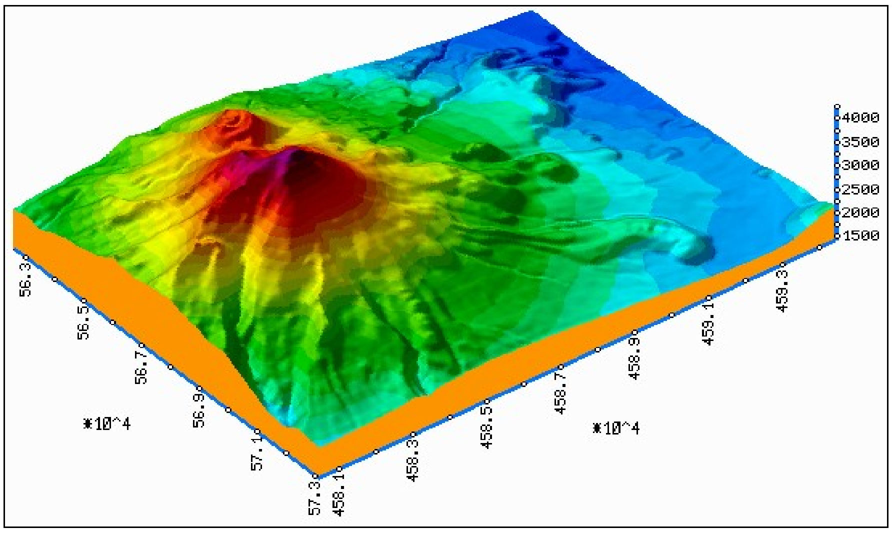


Fig. 5.7: Digital elevation model of mount Shasta in Northern California.

5.8 Construction of model grid

One of the most important applications of SurGe in my profession are projects of geological models of underground gas storages (UGS) in the Czech Republic. The geological models were implemented for these storages:

- Dolni Dunajovice
- Lobodice
- Stramberk
- 9.-11. sarmatian layer in Tvrdonice

SurGe was used for the map creation of both geological surfaces of layers and maps of rock properties such as porosity, permeability or net to gross ratio.

Geological models had to be transferred into a 3D grid for a mathematical model in ECLIPSE (3D three phase model of fluid flow in porous media, see [S10]) which enables to work with so-called corner point geometry (CPG) grids. A corner point geometry grid is formed by general hexahedrons and makes it possible to adapt the grid to the shape of a reservoir structure including curved boundaries, tectonic faults and wedging out of layers.

An important concept in the construction of grid data for corner point geometry is the idea of “coordinate lines”. Coordinate lines are straight lines upon which all the cell corners must lie. Thus if there are NX cells in the x-direction and NY cells in the y-direction, there will be (NX+1)*(NY+1) coordinate lines. Models of the above mentioned storages were created using vertical coordinate lines, but in more general grids they can be off-vertical to coincide with sloping faults (see the next figure).

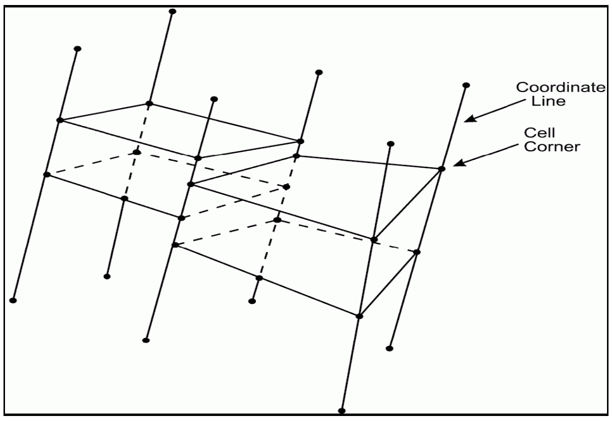


Fig. 5.8a: Construction of corner point geometry grids.

The ECLIPSE digital representation of corner point geometry grid consists of data following two keywords COORD and ZCORN, which have to be included in the input data file. The keyword COORD introduces the data describing coordinate lines and the keyword ZCORN introduces the data describing z-coordinates of cell corners.

To construct a model grid for ECLIPSE, three utilities have been developed:

- EGRID is a simple graphical utility for interactive creation of a model grid horizontal projection consisting of general quadrangles.
- CPG is a console utility generating data for keywords COORD and ZCORN. As an input, the CPG utility uses the horizontal projection of the model grid created by the EGRID utility and surfaces of layers generated by SurGe. The surfaces of layers must be specified (using the suffix convention) in order of increasing depth beginning from the most top layer, and the top surface of each layer must be specified as the first. There may be interspaces between layers, which are considered to be impermeable and which are used for modelling of impermeable bands.
- PROP is a console utility generating data for keywords, which describe rock properties, for example PORO (porosity), PERMX (permeability in the x direction), NTG (net to gross ratio), SGAS (gas saturation) and so on. As an input, the PROP utility uses the horizontal projection of the model grid created by the EGRID utility and maps of layer properties created by SurGe. Layers must be specified (using the suffix convention) in order of increasing depth beginning from the most top layer.

An example of a complicated ECLIPSE model grid with faults, which was created using named utilities, is in the next two figures. The model consists of ten layers having very heterogeneous properties – porosity and permeability. As the geological model shows, there are "channels" with high porosity and permeability in the bottom layers where preferential fluid flow can be expected.

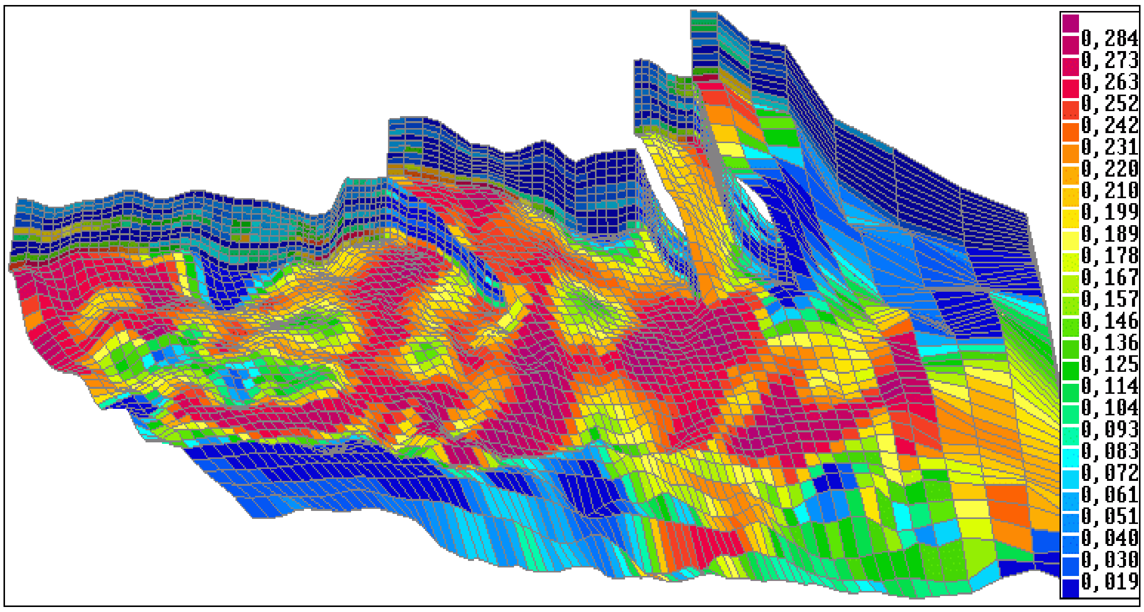


Fig. 5.8b: Porosity of a geological model.

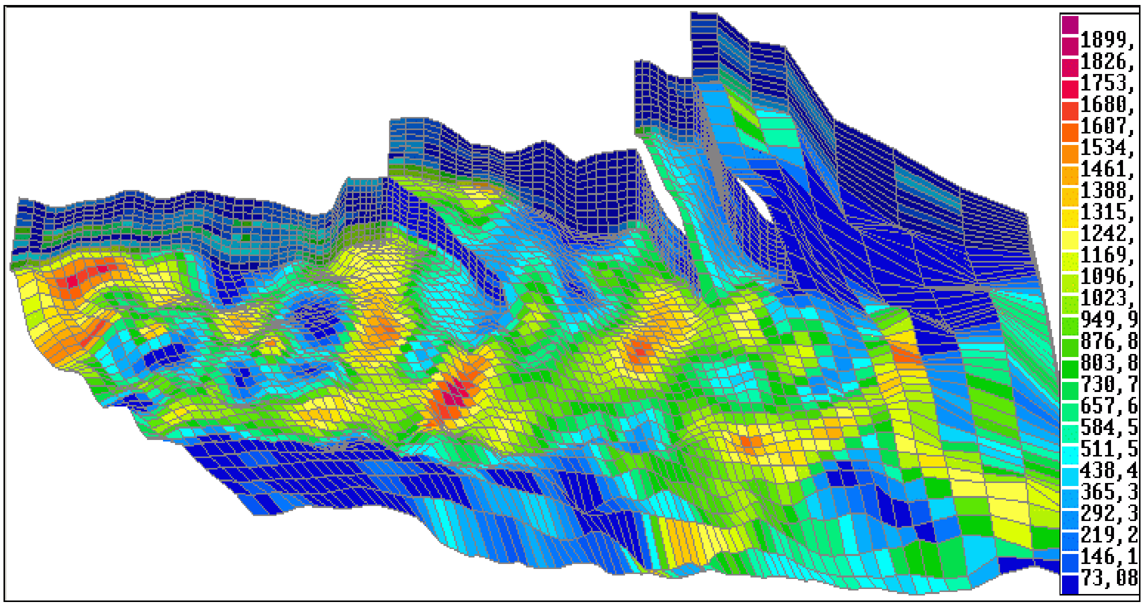


Fig. 5.8c: Permeability of a geological model.

The previous two figures are hard-copies of screens displayed by GRIDV, an OpenGL application written as a post-processor for viewing and checking the ECLIPSE model results.

Conclusion

The most important result of this work is the design and especially the computer implementation of the new interpolation / approximation method ABOS for digital generation of surface passing through *Z* coordinates of points irregularly distributed in 3D Euclidean space. In contrast to sophisticated interpolation methods such as Radial basis functions, Kriging or Minimum curvature, the ABOS method does not fulfil any mathematical formulation such as linear combination of basis functions, statistical formulation of the best linear unbiased estimate or requirement of minimal curvature of the resulting surface. Instead, it provides tools for modelling the resulting surface based on numerical tensioning and smoothing enabling to create a broad range of surface shapes and to accommodate the resulting surface according to the user's conception. Moreover, the ABOS method makes it possible to solve large problems without the necessity to search points in a specified surrounding of the grid nodes and without the necessity to solve a system of equations. As for computational speed, ABOS is up to twenty times faster than methods based on solving equation systems, for example the Kriging method.

The computer implementation SURGEF.EXE and namely the implementation of user graphical interface SurGe extends the applicability of the ABOS method and enables to:

- create and manage projects containing many maps
- easily change interpolation parameters and to experiment with surface shapes
- filter input data
- transform coordinates of map objects (points, faults, polylines, boundaries)
- digitise map objects using mouse and keyboard
- model discontinuity in the generated surface
- perform mathematical calculations with surfaces
- blank the grid nodes outside the boundary
- double the grid using cubic polynomial interpolation between grid nodes
- output z coordinates of the generated surface at any set of points from domain
- calculate isolines and display them
- display a raster colour map and 3D view of the resulting surface
- display a shadowed colour map and shadowed 3D view
- define and display cross-sections through several surfaces
- compute and display gradient lines
- solve special tasks using special procedures such as
    - wedging out of layers
    - preserving extrapolation trend in areas without data
    - direct conversion of seismic reflection times into the structural depth of layer boundary
    - creating layer thickness maps and volume calculations

One of the most important applications of SurGe in my profession are projects of geological models of underground gas storages (UGS) in the Czech Republic. The geological models were implemented for storages Dolni Dunajovice, Lobodice, Stramberk and 9th – 11th sarmatian layer in Tvrdonice, where in all cases the ABOS method was applied for the map creation of both geological surfaces of layers and rock properties such as porosity, permeability and net to gross ratio.

Furthermore, the method was used for model creation of aquifer structures while processing remediation projects in localities CHZ Sokolov, Stráž pod Ralskem and Licoměřice.

After more than three-years of publication through the medium of Internet WWW pages (see [13]) and through the medium of more than 300 software distributors (see ***List of selected SurGe distributors*** in ***References***), the ABOS method and SurGe software has proven wide applicability. Now thousands of people worldwide use the software not only in traditional fields as reservoir engineering, geology and geophysics, but also in hydrology, geodesy, archeology, bathymetry, gravimetry, genetics, neurology, electronics, aerodynamics or nuclear physics; some people use SURGEF.EXE as an interpolation engine in their own mapping software.

References

Software and data source

[S1] Generic Mapping Tools, http://gmt.soest.hawaii.edu/

[S2] Golden Software, http://www.goldensoftware.com/products/surfer/surfer.shtml

[S3] K. L. Clarkson, http://cm.bell-labs.com/cm/cs/who/clarkson/2dch.c

[S4] LandSerf, http://www.landserf.org/

[S5] Global Mapper, http://www.globalmapper.com/

[S6] ESRI Shapefile, http://www.esri.com/library/whitepapers/pdfs/shapefile.pdf

[S7] Inno Setup, http://www.jrsoftware.org/

[S8] Atlas DTM, http://www.atlasltd.cz/

[S9] USGS, http://rockyweb.cr.usgs.gov/nmpstds/demstds.html

[S10] Schlumberger, http://www.slb.com/content/services/software/reseng/index.asp

[S11] Mathworld, http://mathworld.wolfram.com/VoronoiDiagram.html

List of selected SurGe software distributors

http://www.simtel.net/product.php[id]77464[sekid]0[SiteID]simtel.net

http://www.downloadplanet.net/info37162.htm

http://www.fileboost.net/directory/education/science/surge/029209/review.html

http://www.softpicks.net/software/SurGe-8048.htm

http://www.softpedia.com/get/Science-CAD/SurGe.shtml

http://www.yankeedownload.com/software/surge-ilicp.html

http://www.geologicresources.com/software.html

http://www.forestpal.com/Toolbox_mappers.html#surge

http://www.vterrain.org/Elevation/contour.html

SurGe WEB pages

http://surgeweb.sweb.cz/surgemain.htm

http://surge.czweb.org/surgemain.htm

Bibliography

[1] W.H.F. Smith, P.Wessel, *Gridding with continuous curvature splines in tension*, Geophysics, Vol.55, No.3, p.293-305, 1990

[2] D.G. Krige, *A statistical approach to some basic mine valuation problems on the Witwatersrand*, Journal of the Chemical, Metallurgical, and Mining Society of South Africa, vol. 52, pp. 119-139, Dec. 1951

[3] C. V. Deutsch, A. G. Journel, *GSLIB. Geostatistical Software Library and User's Guide*. Oxford University Press, New York, NY, 1998

[4] T. H. Mayer, *The Discontinuous Nature of Kriging Interpolation for Digital Terrain Model*, Cartography and Geographic Information Science, Volume 31, Number 4, 2004, pp. 209-216(8)

[5] C. de Boor, *A practical guide to splines*, Springer-Verlag., 1978

[6] Ch. Yang, S. Kao, F. Lee, P. Hung, *Twelve Different Interpolation methods: A Case Study of Surfer 8.0,* Geo-Imagery Bridging Continents, XXth ISPRS Congress, 12-23 July 2004 Istanbul, 2006

[7] N. Sukumar, *A Note on Natural Neighbor Interpolation and the Natural Element Method (NEM),* Theoretical and Applied Mechanics, Northwestern University, 1997

[8] H. Mitasova, L. Mitas, *Interpolation by Regularized Spline with Tension: I. Theory and Implementation*, Mathematical Geology, 25, 641-655, 1993

[9] I.C. Briggs, *Machine Contouring using minimum curvature*, Geophysics, 39, 39-48, 1974

[10] M. Dressler, *Computation and presentation of surface in geology*, Zemní plyn a nafta, 40, 4, 317-323, 1996

[11] J. Hora, M. Dressler, V. Wasserbauer, *Model Diamo for ground-water flow and transport of chemicals*. Proceedings from Environmental Statistics and Earth Science, Brno, 1996.

[12] V. Onderka, M. Dressler, *STRAZ BLOCK GROUNDWATER POLLUTION - PHASE I*, GeoGas Brno, 1995.

[13] M.Dressler, *Approximation / interpolation of surfaces based on numerical tensioning and smoothing,* http://www.sweb.cz/M.Dressler/ABOS.htm