Oil & Gas Science and Technology – Rev. IFP Energies nouvelles **75**, 62 (2020)

https://doi.org/10.2516/ogst/2020057

Available online at:
ogst.ifpenergiesnouvelles.fr

Advanced modeling and simulation of flow in subsurface reservoirs with fractures and wells for a sustainable industry
S. Sun, M.G. Edwards, F. Frank, J.F. Li, A. Salama, B. Yu (Guest editors)

REGULAR ARTICLE

OPEN ACCESS

Fast interpolation method for surfaces with faults by multi-scale second-derivative optimization

Michel Léger and Vincent Clochard*

IFP Energies nouvelles, 1 et 4 avenue de Bois-Préau, 92852 Rueil-Malmaison Cedex, France

Received: 5 March 2020 / Accepted: 10 July 2020

Abstract. We present a smooth surface interpolation method enabling to take discontinuities (*e.g.* faults) into account that can be applied to any dataset defined on a regular mesh. We use a second-derivative multi-scale minimization based on a conjugate gradient method. Our multi-scale approach allows the algorithm to process millions of points in a few seconds on a single-unit workstation. The interpolated surface is continuous, as well as its first derivative, except on some lines that have been specified as discontinuities. Application in geosciences are numerous, for instance when a structural model is to be built from points picked on seismic data. The resulting dip of interpolation extends the dip of the input data. The algorithm also works if faults are given by broken lines. We present results from a synthetic and real examples taking into account fault network.

1 Introduction

The picking of surfaces, isochrones or faults, is difficult and uncertain because of low signal to noise ratio of surface seismic or other reasons, so that most often only a partial picking can be achieved. However, for some geoscience applications (Schneider, 2002; Perrin and Rainaud, 2013), a complete picking is required through these areas. An interpolator is thus needed with following characteristics: it has to be fast, able to process millions of data points, adaptable to all kinds of dataset (Fig. 1), and it has to preserve the dip of the data (if any). Moreover, the interpolator has also to take faults into account.

Numerous interpolation techniques (Awange *et al.*, 2018) like polynomial interpolations (Léger, 1999) are not compatible with the above requirements, also spline interpolations are currently used in the scope of Computer Aided Design (CAD) to define object representation and to perform various manipulation of surfaces. Nevertheless these representations require mandatory conditions like, global and local shape control using control point of a surface. As well, to insure a continuity at the junction of two patches, it is necessary to define a smoothing quality factor around control points, usually first order with tangent and second order with curvature. Another difficulty is also to define control points outside of the surface to limit the edge effects, so the control points are not located on the surface.

We have mainly two kinds of models called Bezier- and B-spline surfaces (Bézier, 1977, 1987), both are used according to the chosen application. It exists also two implementations of these models rational or non-rational. In this last case a control point of a Bezier model has a global influence and with B-spline it is only a local influence. With the rational implementation, we speak about Non Uniform Rational B-Spline (NURBS), which consists to associate at each control point a weighting factor to give more or less influence in the representation (Rogers and Earnshaw, 1991). Another aspect is due to the degree of a rational Bezier surface defined by the number of control point minus one, and the degree of a B-spline surface which could be chosen, weak to have more local control on the surface. One can think to the user difficulties of choosing the best parameters to build a NURBS surface and to avoid edge effects to interpolate a surface with discontinuities.

Another approach is based on a local optimization of a parametric surface (Hjelle and Dæhlen, 2005). It is a local least-square approximation with a regularization term. They used hierarchical triangulations iteratively refined if the data are irregular, typically contour lines maps. More precisely, the process begins with a regular square mesh and using each square centers, it makes four triangles. Then, considering the middle of the longest edge it makes two triangles instead of one. This kind of dichotomy only occurs if the data are sufficiently numerous and the process is iterative, so they work with a varying spatial step. The main difficulty of the method is how to weight the smoothing term. Another method uses a combination of Radial Basis Functions (RBFs) and Moving Least Squares (MLS) in Ohtake *et al.* (2004). Other authors used hierarchical bases using B-splines and spline-(pre)wavelets with a regularization parameter (Castaño *et al.*, 2009). At each

* Corresponding author: vincent.clochard@ifpen.fr

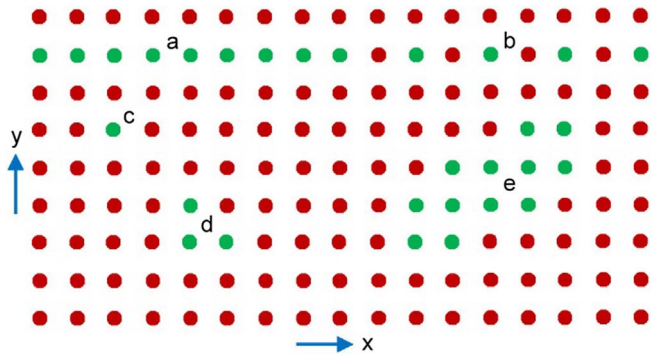

Fig. 1. Any dataset can be interpolated on a regular 2D-grid (all green and red points), taking into account random distribution of given values at green points. (a) These green points could represent lines, (b) dotted lines, (c) single points, (d) small groups of points, (e) large datasets.

stage a least squares approximation of the data is computed which is quite efficient to approximate large amount of scattered data points (up to 300 000). More recently, huge amount of data come from Light Detection And Ranging (LiDAR) for the derivation of Digital Terrain Models (DTMs), a least squares Compactly Supported Radial Basis Function (CSRBF) interpolation method is quite efficient and four times faster compared with ordinary kriging (Chen *et al.*, 2018a, 2018b).

We present here a fast multi-scale interpolator able to work with a very limited choice of interpolation parameters and also to take into account any discontinuities. The main differences of our interpolation with last cited work is the use of a constant spatial step at each iteration and also the use of a single cost-function defined overall. One can note our cost-function has a single minimum as soon as the data have at least three points (not aligned).

In the following the interpolator considers only vertical faults but its extension to inclined faults is straightforward (Léger, 2016).

This interpolator relies on the minimization of the second derivative, using a conjugate gradient (Hestenes and Stiefel, 1952). This leads to a good solution but the computing cost is too large because large unknown regions require a high number of iterations. To get a good convergence for a large number of unknowns with a small number of iterations, a good initial model is required. This is achieved by decimating the data point set with a double step in X and a double step Y (computational time divided by 4) and the number of iterations being doubled. As a whole, computation time is divided by 2. Reiteration of this decimation answers the issue of getting the initial model. The reiteration stops when the number of iterations is greater than the number of unknowns, the minimization of the cost function being thus complete. This explanation goes from the fine grid to the coarser grids, but the computation goes from the coarser grids to the fine grid.

Faults are introduced, *via* broken lines, along which the calculus of all derivatives crossing them are excluded. This very simple trick allows interpolation of surfaces presenting discontinuities such as faults, which is a great improvement from a geological point of view. The computation time remains the same, whatever faults are present or not.

The key point of this approach is that 100 iterations on the final model are sufficient to converge in case of simply or moderately complex models, whatever the size of the model is, and whatever faults are present or not.

In Section 2, we explain the method. In Section 3, we illustrate the method on different synthetic examples without faults. Section 4 is devoted to synthetic examples with faults. A real example is presented in Section 5.

2 Method

2.1 Principle

We consider a rectangular regular mesh and the integer coordinates (i, j) discretize the coordinates X and Y. The step in i and j are unit, but the trace distance in meters is given to the program to manage possible anisotropies between X and Y. The data points lie on the mesh, and the unknown points also. To interpolate some given points, we have chosen to compute a surface that goes through all given points and to minimize its second derivative with the conjugated gradient method, which is simple to implement. However for some datasets, the number of iterations can be close to the number of unknown points, making the algorithm $O(N^2)$, which this is too expensive for many applications. The aim of this paper is to remove this disadvantage.

A solution to obtain a quick optimization is to initialize the model optimization by a model very close to the solution. But how to get a such initial model? The solution relies on the fact that optimizing a model with a number of i and j twice less than the original ones, and doubling the number of iterations makes overall twice less computations (Jespersen, 1984). Thus, how to get a new good initial model? Just iterate! This reiteration stops if the conjugate gradient converge in a number of iterations which is greater than the number of unknowns. This is true because the second derivative is a linear function of the unknowns, so the cost-function is a quadratic (see Ref. "Conjugated Gradient Method"). In this case, it is the first inversion, and the result will be independent of the initial model. In practice, the overall shape is obtained on the coarse grids, especially the first, and next finer and finer details on other grids.

This multi-scale strategy makes the algorithm $O(N)$ instead of $O(N^2)$. The complexity of the data layout, and especially of the fault network, increase the cost, that is, we get $O(aN)$ with greater a. For instance, $a = 100$ is a value adequate for simple data, but $a = 1000$ or more could be necessary for more complicated data. Once the data set and the faults are given, the cost is really in $O(N)$ in case of mesh refining.

2.2 Second derivative measurement

Starting from a regular mesh of points, it is easy to construct a C^2 piecewise polynomial surface, such as splines. *Via* this possibility, we will use the slope and the second derivative as well. Nevertheless, these first and second

derivatives (see Sect. 2.7) are specific to our model and depend of its characteristics.

In order to compute the three kinds of second derivatives, we simply use the following formulas of a function f (see Ref. "Finite Difference"). With our convention parameters, the expression for the second derivative d_{ii} in i, where the h_i is the step in i and (i, j) is the position of the point on the regular mesh is given by (1):

$$h_i^2 d_{ii}[f(i,j)] = f(i+1,j) - 2f(i,j) + f(i-1,j). \quad (1)$$

The expression for the second derivative d_{jj} in j, where the h_j is the step in j and (i, j) is the position of the point on the regular mesh is given by (2):

$$h_j^2 d_{jj}[f(i,j)] = f(i,j+1) - 2f(i,j) + f(i,j-1). \quad (2)$$

The expression for the second derivative d_{ij} in i and j could be chosen as we like. The most popular equation is given by (3), completely symmetric and easy to differentiate, but not very compact:

$$4h_i h_j d_{ij}[f(i,j)] = f(i+1,j+1) - f(i+1,j-1) - f(i-1,j+1) + f(i-1,j-1). \quad (3)$$

Another possibility is less symmetric and mixes two schemes, one positive in i and j, the other negative in i and j, but the seven terms are a disadvantage for differentiation (4):

$$2h_i h_j d_{ij}[f(i,j)] = f(i+1,j+1) - f(i+1,j) - f(i,j+1) + 2f(i,j) - f(i-1,j) - f(i,j-1) + f(i-1,j-1). \quad (4)$$

We have chosen for the cross-term d_{ij} a no-symmetric solution (5), with "positive in i and j" scheme, because it is compact and easy to differentiate:

$$h_i h_j d_{ij}[f(i,j)] = f(i,j) + f(i+1,j+1) - f(i+1,j) - f(i,j+1), \quad (5)$$

whatever the scheme is, this cross-term must be taken into account, as Figure 2 (bottom part) shows its influence.

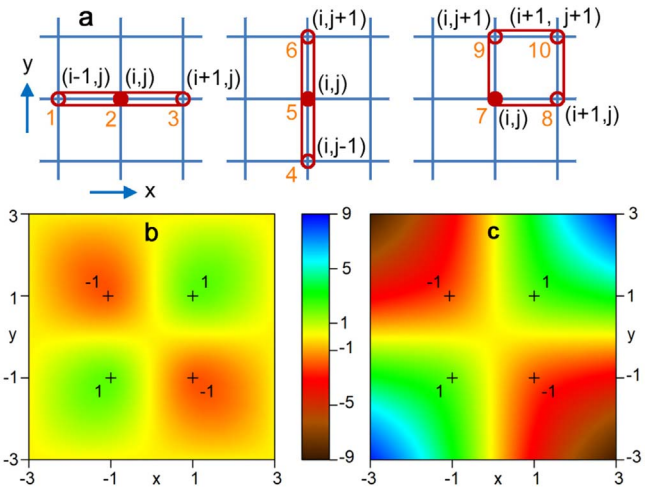

Fig. 2. Top part (a) illustrates the three second derivatives of a function at the filled red point (i, j). The numbers in orange represent the components of the gradient, for the point (i, j). Bottom part shows the influence of the second derivative (using our four-term scheme) in the interpolation result: (b) with and (c) without, both interpolations using the same initial data (value of $+1/-1$ at the black crosses). We see on (b) a very smooth result by the edge, whereas the corners are very far from the data in (c). The ruled surface displayed in (c) is a hyperboloïd.

2.3 Cost function

Cost function Q is the squared L^2 norm of the second derivatives:

$$2Q = Q_{II} + 2Q_{IJ} + Q_{JJ}, \quad (6)$$

with,

$$Q_{II} = \sum_{N_{II}} \left(h_{ii}^2 d_{ii}[f(i,j)]\right)^2, \quad (7)$$

where N_{II} is the set of points with at least an unknown in the triplet in i,

$$Q_{IJ} = \sum_{N_{IJ}} \left(h_i h_j d_{ij}[f(i,j)]\right)^2, \quad (8)$$

where N_{IJ} is the set of points with at least an unknown in the quadruplet, and,

$$Q_{JJ} = \sum_{N_{JJ}} \left(h_{jj}^2 d_{jj}[f(i,j)]\right)^2, \quad (9)$$

where N_{JJ} is the set of points with at least an unknown in the triplet in j. The sets N_{II}, N_{IJ} and N_{JJ} are usually somewhat different. These cost functions are computed on several grids, at each scale.

Cost function Q is made of only one criterion, so the result will be independent of any overall multiplicative factor such as uncertainties (Foster and Evans, 2008).

Since the cost-function at a given point depends on few points, the gradient is computed by finite differences.

2.4 Faults

Faults correspond to discontinuities that are defined along broken lines. The shape of these faults is defined at the beginning, whatever the shape of the surface that will evolve during the optimization. In others words, faults will be vertical and the same fault model will be used at all scales. These faults are used as such, without any extrapolation which means the algorithm will introduce discontinuities exactly where the customer has specified.

The principle for handling faults is to remove the minimization of the second derivatives crossing the faults. Figure 3 shows a region where all points are unknown. The displayed triplets in dotted red or blue show second derivatives which are removed in the cost function, due to the presence of the fault in green. Doing so, the two sides

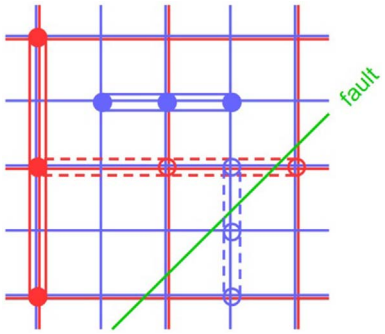

Fig. 3. The red mesh and the blue mesh represent two scales of modelling. The triplets are second derivatives which are conserved in full lines, and removed in dotted lines if they cross the fault.

of the fault become disconnected. This trick works for any scale in the multi-scale inversion.

2.5 Multi-scale data and results

Data are given on the fine regular grid. But they have to be downscaled fully automatically on the coarser ones while keeping the mean and also the slope of the given points, if any. Like the interpolated values are scalars which could represent anything we use the word slope in a general meaning. If these points are on a plane, whatever their distribution, then the result should belong to such a plane. This work must be done for each scale. It is possible that a group of different points on finer grids becomes isolated on coarser grids.

The results obtained on a coarser grid must be interpolated by a bilinear method to initialize the next finer grid. The problem is simple if there is no fault, a bilinear interpolation is sufficient to guess the lacking points.

If there is a fault in the vicinity of the point, the bilinear interpolation will fail because the error can be as large as the throw of the fault. In this case, the conjugated gradient will finish to solve the problem, by making a bulge with increasing size and decreasing amount, but this is very long to do so. For this reason, we need to accelerate it by a "pre-conjugated gradient" phase. Before the optimization, this phase consists in estimating the value of points closed to the fault by using points on the same side of the fault but further away. The aim is to completely decouple the value of a point from any value on the other side of a fault. The "pre-conjugate gradient" phase is the key for the quickness of the procedure.

2.6 Optimization

The interpolator is defined as the function $f(i, j)$ which minimizes the L^2 norm of the second derivative, with arbitrary data, i representing the abscissa X and j the ordinate Y. On a rectangular grid of N points, the number of unknowns may vary from 1 (a single point is given) to $N - 1$ (all points are known except one). All intermediate situations are allowed.

Table 1. Computation time as a function of the number of inversions, for 340 000 unknowns. We prefer 2 s (multi-scale) than 1 h (single scale).

Number of inversions	Number of iterations	Computation time (s)
1	400 000	3611
2	42 240	553
3	5300	78
4	670	11
5	100	2

The multi-scale situation has consequences on the inversion. A small set of given points may show an average slope which will influence the result on the finer meshes. On coarser meshes, this same small set of points will become a single point, and no slope can be attributed to this point. This means that the slope has a meaning only on the finer scales, and since the number of iterations is reduced, the zone of influence of it will be also reduced. This is different from a single-scale inversion where this small set will have long term influence up to others data points.

The implementation of the conjugate gradient is very simple. The customer has to choose the number of iterations on the last inversion, and the program computes the numbers of iterations for the others inversions. The ratio between the number of unknowns and the number of iterations on the last optimization decide how many scales are necessary. The formula is:

$$N_O = 1 + \text{ceiling}(\ln_2(N_U)/3 \ln_2(N_I)), \qquad (10)$$

with $\ln_2(\cdot)$ being the base 2 logarithm, *ceiling* the function which associates the integer immediately superior to a real value, N_O the number of optimizations, N_U the number of unknowns, and N_I the number of iterations on the last model. To increase (respectively decrease) the number of scales, the simplest is to divide (respectively multiply) by 8 the number of iterations on the last model.

Consider an example with $N_U = 340\,000$. Optimization in one pass, leads to $N_O = 340\,000$, and a lot of computations: 3611 s on a Dell T3610 workstation. If we optimize in five steps with 100 iterations on the last model, then the computation time decreases to 2 s. Table 1 shows the relationship between the number of iterations on the last model and the computation time.

Figure 4 shows the scheme of the algorithm. In violet, we see the number of unknowns in a logarithmic scale, with a factor equal to 4 (2 for X and 2 for Y). The number of inversions is in blue. In red, the number of iterations with a factor of 2. In case of one inversion and the computation time is one hour, whereas it is only 2 s if 5 inversions are used.

2.7 Smoothness of the interpolated surface

In practice, we do not see in the results the difference with a continuous slope surface. In Appendix, we show that the optimization of second derivatives does not lead to a

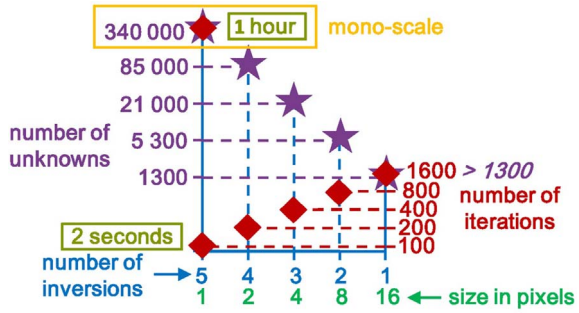

Fig. 4. Scheme of the algorithm.

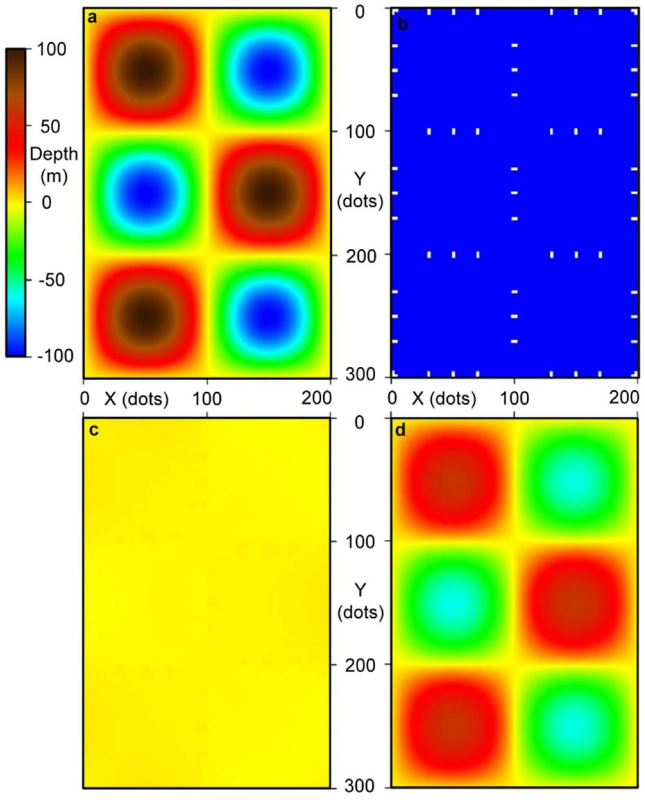

Fig. 5. (a) True model varying between −100 and +100. (b) Unknowns are in dark blue and bi-points give dips, with small values included in the interval [−3.2, 3.2]. (c) Result of the inverse distance weighting, which preserves the data interval [−3.2, 3.2] with a yellow color everywhere. (d) Result of the multi-scale algorithm which preserves slopes and gives [−60, 60] overall range.

C^2 resulting surface, but to an almost C^1 surface because of the Sobolev inequality.

3 Synthetic examples

This section shows applications on several synthetic examples which develop the characteristics of the interpolator, first without faults, and second with faults.

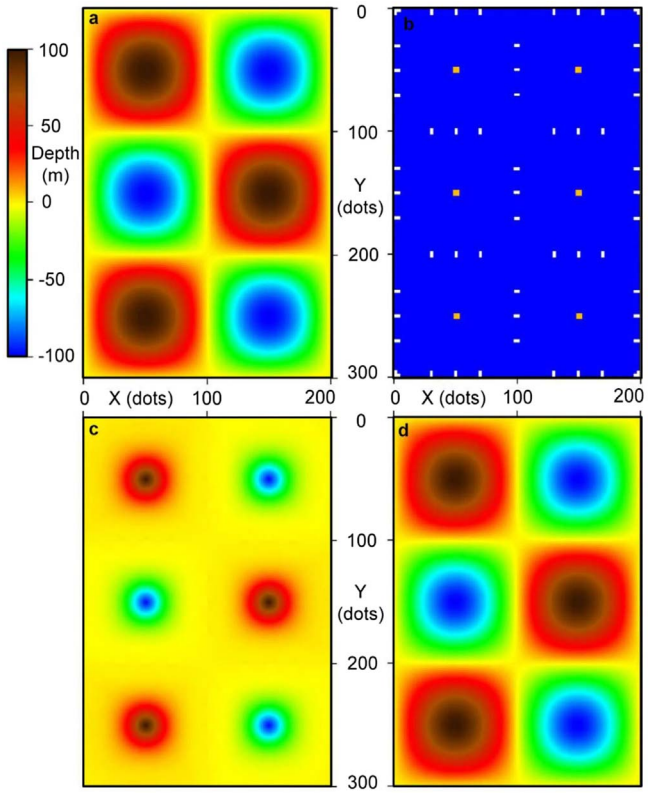

Fig. 6. (a) Same true model. (b) Six points in orange are added at minimum or maximum values. (c) Result of the inverse distance weighting, with dip discontinuity at the data points. (d) Result of the multi-scale algorithm with an maximal error less than 6%.

3.1 The continuity of the slope

Figure 5a presents the product of two sinusoids, one in X, the other in Y. The size of the mesh is 200 by 300 dots and the height varies from −100 to 100 m. We compare our interpolation method with the inverse distance weighting (see Refs. "Inverse Distance Weighting"; Lu and Wong, 2007). For simplicity and robustness, the inverse distance weighting has the advantage to preserve the minimum-maximum range between the data and the result. In general, this is not true with our interpolator because the slope is continuous at a maximum or minimum value, the dip goes down in some direction and necessarily dip goes up in the opposite direction.

Figure 5b shows white rectangles which are pairs of points oriented along the length. Dark blue represents the unknown. These points are very close to zero since their maximum is less than 3.2 whereas the overall interval is [−100, 100].

Figure 5c shows the result of the inverse distance method, which preserves the data interval [−3.2, 3.2]. On the contrary, we can see on Figure 5d that our interpolator takes into account the slopes themselves, and the overall interval [−60, 60] is much closer to the theoretical [−100, 100].

We have added data points at minimum and maximum values as shown in Figure 6. The result of the inverse

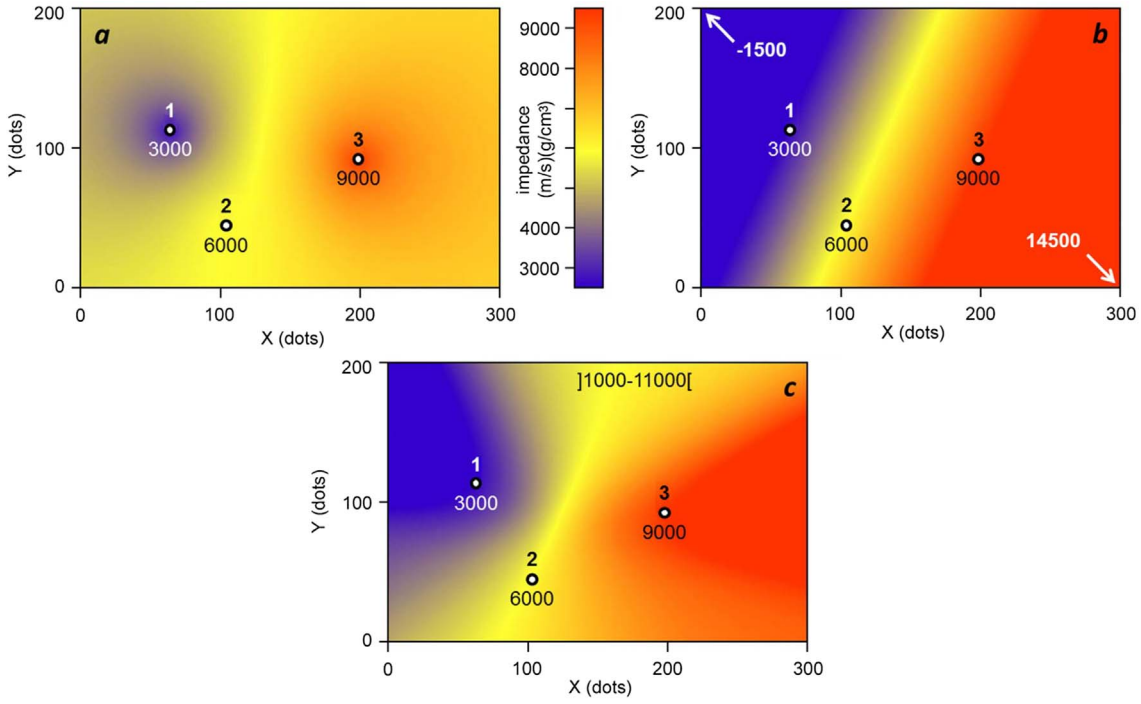

Fig. 7. (a) Inverse distance weighting seems correct since the minimum-maximum range is preserved from data to result. (b) Our interpolator is not correct since the minimum is negative. (c) Anamorphosis makes the result satisfactory inside the interval]1000, 11 000[(m/s) (g/cm^3).

distance weighting appears in Figure 6c with discontinuous slope at all data points, especially at orange points. Our interpolator (Fig. 6d) yields a continuous slope everywhere, even at data points, and the maximal error is less than 6%.

3.2 Anamorphosis

The inverse distance property to preserve the minimum-maximum range may be comfortable because there no risk to get aberrant values. We have seen in the previous paragraph that this is not the case for our interpolator. It may happen that some values become negative and this might be an issue for properties like velocity. Figure 7a shows three wells with extremely contrasted acoustic impedance values, ranging from 3000 to 9000 (m/s) (g/cm^3). The result of the inverse distance (Fig. 7a) shows discontinuities at wells but the overall shape is satisfactory. Nevertheless we can notice a "bubble effect" around the points 1, 2, 3 and if we consider them as well-measurements it could have negative impact in model building (Sams and Carter, 2017) and further for geophysical applications like reservoir characterization. This is not the case of our interpolator (Fig. 7b) since the interpolator goes down to −1500 (m/s) (g/cm^3), which is not acceptable for acoustic impedance for instance.

To solve this problem, we need a technique that modifies the values outside the interpolation. The anamorphosis is adequate for that purpose. It requires a change of variables before the interpolation, which makes wishable interval] L_1, L_2 [become]$-\infty$, $+\infty$[, and the inverse variable change from] $-\infty$, $+\infty$ [to] L_1, L_2 [after the interpolation. With $M_{12} = (L_1 + L_2)/2$ and $D_{21} = L_2 - L_1$, the formula of the forward anamorphosis is:

$$g_d(i,j) = M_{12} + \frac{D_{21}}{\pi} \tan\left(\frac{\pi}{D_{21}}(f_d(i,j) - M_{12})\right), \quad (11)$$

and for the backward anamorphosis is:

$$h_i(i,j) = M_{12} + \frac{D_{21}}{\pi} \tan^{-1}\left(\frac{\pi}{D_{21}}(g_i(i,j) - M_{12})\right). \quad (12)$$

Functions f_d and g_d concern the data, whereas g_i and h_i concern the interpolation result. Limits L_1 and L_2 should be not too closed to the data. In Figure 7c, $L_1 = 1000$ and $L_2 = 11\,000$ (m/s) (g/cm^3), have been chosen and the result remains in this interval.

Note that the anamorphosis limits are not necessarily constant, they can vary with i and j. In such a case, quantities M_{12} and D_{21} become M_{12} (i, j) and D_{21} (i, j).

Finally, if we compare Figure 7a and the final result of our interpolator (Fig. 7c), it does not include the "bubble effect" and respect the physical range of limits of the considered parameter to interpolate.

4 Synthetic examples with faults

We now introduce faults in the models aiming to show the efficiency of the "pre-conjugate gradient" phase, which is critical for the accuracy and the quickness of the

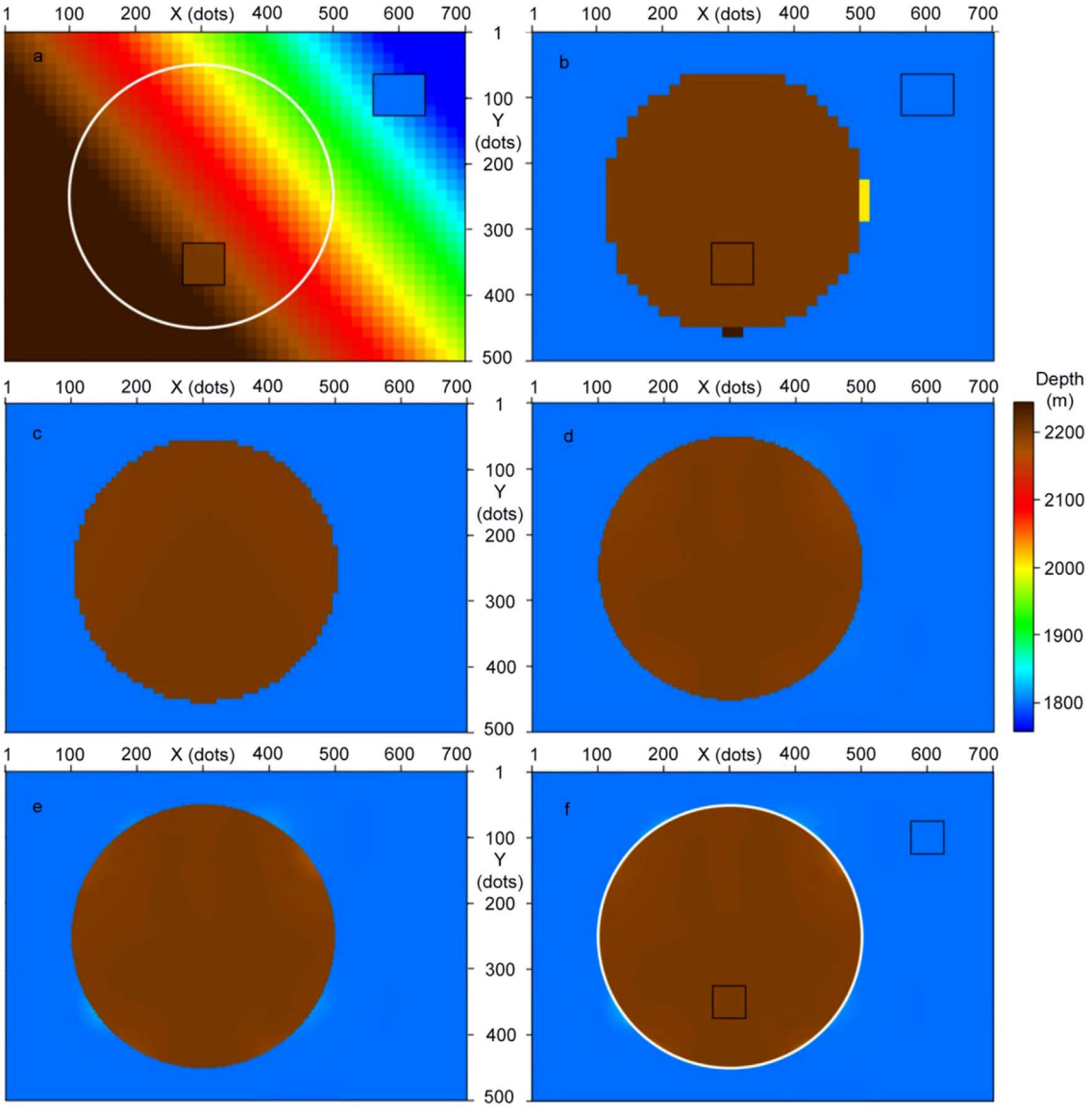

Fig. 8. (a) Initial model. (b) Result of the first iteration with 1300 parameters and 1600 iterations. (c) Result of the second iteration with 5300 parameters and 800 iterations. (d) Result of the third iteration with 21 500 parameters and 400 iterations. (e) Result of the fourth iteration with 85 000 parameters and 200 iterations. (f) Result of the fifth, and last, iteration with 340 000 parameters and 100 iterations. The L^2 norm of errors is 3.5 m for a 400 m difference in the data.

interpolator. Another aim is to test the behavior of the algorithm at the ends of faults.

4.1 Circle

This example has been designed to show the independence between the result and the initial model. The white circle, representing a closed discontinuity (Figs. 8a and 8f), is going to show how the interpolator manages all orientations of the fault. In this example we have only two given set of values (two little black-squares with brown and blue colors), one inside the white circle and the other outside.

The dataset consists of two horizontal squares, one at 1800 m depth, the other at 2200 m depth. The size of these squares is 50 dots, and the size of the unknown domain is 700 by 500 dots. Figure 8a shows the initial model which is the linear regression of the data. Because the same palette is chosen, the larger range saturates the dark brown and dark blue. In Figure 8b, the result of the first iteration has nothing to be compared with the initial model, and the two constant zones at 1800 and 2200 m are clearly found. The number of iterations of this first inversion is 1600 and the number of unknowns is 1300. Figure 8c shows the result of the second inversion, with some more details. Figures 8d and 8e show the results of the third and fourth iterations. Figure 8f shows the final result for the fifth inversion. For a difference of 400 m in the data, the L_2 deviation to the theory is less than 3.5 m, that is less than 1%.

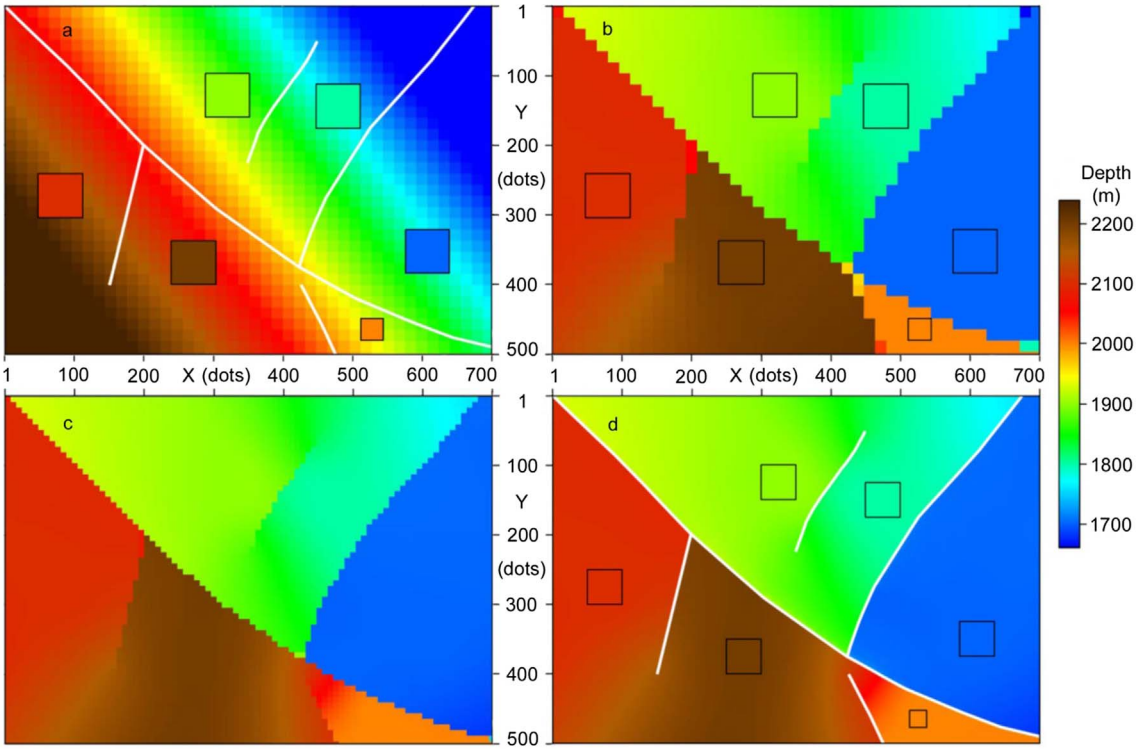

Fig. 9. (a) Initial model of the first model. (b) First model with 1300 parameters and 1600 iterations. (c) Second model with 5300 parameters and 800 iterations. (d) Final result. Faults are white and given points inside black squares.

This synthetic case study demonstrates the efficiency of our interpolator to take into account all fault-orientations and even compartment (a closed limit). In the final interpolation result (Fig. 8f), we have an uniform brown color inside the white circle and an uniform blue color outside.

The overall multi-scale computation time on a Dell T3610 workstation is only 2 s for 100 iterations on the last model, whereas a single direct inversion on the full 340 000 dots needs 3600 s. This is the deciding advantage of the multi-scale technique.

4.2 Five faults example

This example with horizontal data sets at different levels aims to illustrate the continuity of dip.

One square corresponds to 25 × 25 dots, and the others to 50 × 50 dots. For each square, including the interior, all points have the same height, and the interpolator makes dip continuous from the data. In Figure 9a, blue data are 1700 m depth, blue–green 1800 m, green 1900 m, orange 2000 m, red 2100 m and brown 2200 m. The interpolation result is shown in Figure 9b. Note than the flat orange zone is disconnected from the others. Figures 9c shows the second inversion result. The orange zone becomes connected the brown zone. Figure 9d shows the final result. The computation time is 2 s on Dell T3610 workstation for five inversions and with 100 iterations on the last model. We see that the overall shape of the solution is founded at the end of the first inversion, the others iterations settling finer and finer details. One can notice, especially with the

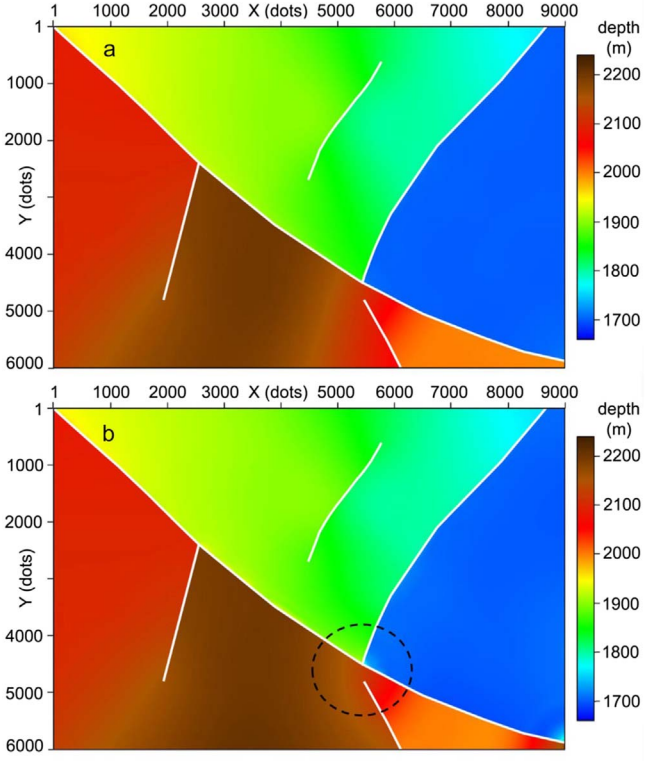

Fig. 10. (a) Result of a 54 000 000 points interpolation with 50 iterations on the eighth and last model. (b) Result with only three iterations on the 10th and last model.

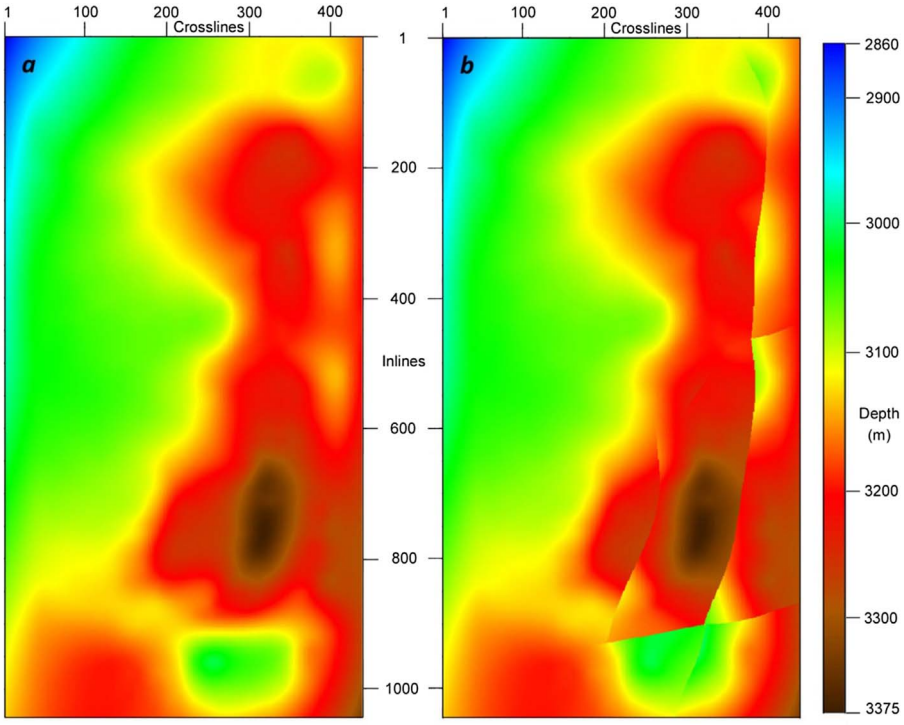

Fig. 11. We see the result of the interpolation of Alwyn data, without faults in (a), with faults in (b).

first inversions (Fig. 9b) the increased size of the square-data (black squares) due to the convolution of the initial black-squares (displayed on Fig. 9d) by the current pixel-size.

Like the previous synthetic example, this more complex case study shows how behaves the interpolator to take into account all fault-orientations. In the final interpolated result (Fig. 9d), we have a compartment with a closed limit (see blue area) and a partial compartment (see orange area). Notice also the intermediate case with the red–brown contrast.

4.3 Big model with 54 000 000 points

This enlargement of the five faults example is designed to show the ability of the interpolator to manage very big meshes. This example shows 9000 points in X and 6000 in Y. Figure 10a shows the result of the eighth inversion with 50 iterations on the last model. One pixel of the initial model corresponds to 65 536 pixels of the final model. This work necessitates 2 min 27 s on a Dell T3610 workstation.

Figure 10b shows the result with only three iterations on the 10th and last model. We can see that the result is not perfect (see inside dotted black circle). One pixel of the initial model corresponds to 262 144 pixels of the final model. The computation time is 23 s.

5 Real case study: Alwyn (north sea)

We present the Alwyn dataset as a real field example. The size of this example is 13 km long and 5.5 km wide, with steps of 12.5 m in X and Y, which represents ~458 000 unknowns. Seismic sections have been picked and the result is a triangulation, the typical distance between points being about 100 m. These triangulation points are projected to the closest points of the regular mesh with the faults being edges.

We have removed all the points closer than 350 m from a fault. This choice is the maximal value which preserve two points along the eastern edge, the two points allowing the dip computation. The reason to remove points close to the fault is also justified by a greater picking-uncertainty due to the signal-to-noise ratio.

We made two interpolations in Figure 11, without the faults in (a), and with the faults in (b). From this real case study, we can see the efficiency of the interpolator to take into account a real discontinuity network (Fig. 11b). We can see clearly a little narrow full compartment (at the right) and partial compartments (at the bottom-right and in the middle). The computation time is quite fast, it takes only 3 s on a Dell T3610 workstation for this ~458 000 unknowns interpolation.

Figure 12 shows the input data which are displayed in (a), and in (b) we see the interpolation difference between (b) and (a) of Figure 11. We can see clearly these differences following the discontinuity network.

At this stage we speak about discontinuities and not fault because we have access only to the (X, Y) position of faults at a given depth, the height of surfaces along them is unknown. With more information the interpolator could be applied many times at different depth in order to build 3D discontinuous surface-model. Many applications in Geosciences could benefit from using *a priori* 3D faulted-model (Mitra *et al.*, 2017).

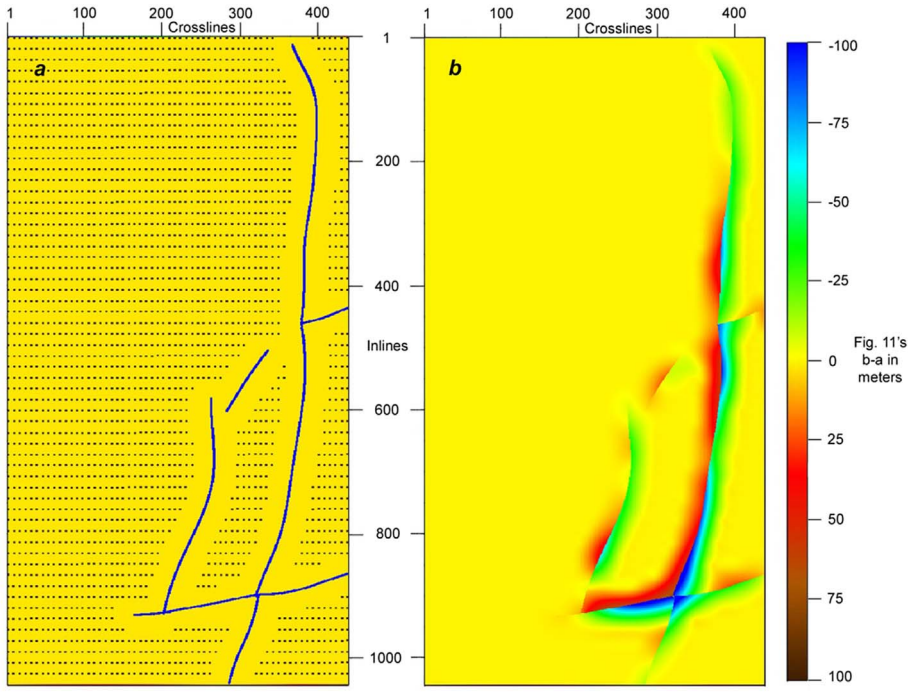

Fig. 12. The data are shown in (a). The difference between (b) and (a) of Figure 11 is shown in (b).

6 Conclusion

During several years, the presented multi-scale interpolation was used in various scope of work from Geosciences without faults to fill sparse information in 2D regular meshes. We bring into focus a multi-scale interpolator which has many advantages. The interpolation is fast because a million of unknowns requires only few seconds to compute. This fact is due to our multi-scale approach with several inversions instead of one, consequently the algorithm passes from N^2 to N. As well, the interpolation follows not only the value of the given data, but also the given dip (if any). From the various processed examples, one can also notice the arbitrary distribution of dataset, from 1 to $N - 1$ known points. Finally, from the last real case study from Alwyn, the interpolation is able to work with any network of fault lineaments. The same interpolator has been used with the same efficiency with a dozen of other unpublished real cases studies, for confidential reasons. Some of them included sixty faults. More generally, the presented algorithm can interpolate any real property available on a regular mesh.

Acknowledgments. We would like to thank *TOTAL* which gives to Energistics Consortium members the set of Alwyn North Field Data used to realize this use case.

References

Awange J.L., Paláncz B., Lewis R.H., Völgyesi L. (2018) *Mathematical geosciences: Hybrid symbolic-numeric methods*, *Springer*. doi: 10.1007/978-3-319-67371-4.

Bézier P. (1977) Essai de définition numérique des courbes et des surfaces expérimentales. Contribution à l'étude des propriétés des courbes et des surfaces paramétriques polynomiales à coefficients vectoriels, *Thèse d'Etat*, Paris VI.

Bézier P. (1987) *Mathématiques et CAO : courbes et surfaces*, Tome 4, Hermès.

Castaño D.D., Jager G., Kunoth A. (2009) Mutiscale analysis of multivariate data, in: *Architecture and Civil Engineering*, K. Gürlebeck, C. Könke (eds), Weimar, Germany.

Chen C., Li Y., Zhao N., Guo B., Mou N. (2018a) Least squares compactly supported radial basis function for digital terrain model interpolation from Airborne Lidar Point Clouds, *Remote Sens.* **10**, 4, 587. doi: 10.3390/rs10040587.

Chen C., Li Y., Zhao N., Yan C. (2018b) Robust interpolation of DEMs from Lidar-derived elevation data, *IEEE Trans. Geosci. Remote Sens.* **56**, 2, 1059–1068. doi: 10.1109/TGRS.2017.2758795.

Claerbout J., Muir F. (1973) Robust modeling with erratic data, *Geophysics* **38**, 5, 826–844.

Conjugated gradient method: https://en.wikipedia.org/wiki/Conjugate_gradient_method.

Finite difference: https://en.wikipedia.org/wiki/Finite_ difference.

Foster M.P., Evans A.N. (2008) Performance evaluation of multivariate interpolation methods for scattered data in geoscience applications, in: *IEEE International Geoscience and Remote Sensing Symposium (IGARSS), Boston, MA, USA*.

Hestenes M.R., Stiefel E. (1952) Methods of conjugate gradients for solving linear systems, *Res. Paper Nat. Bureau Stand.* **49**, 6, 409–436.

Hjelle Ø., Dæhlen M. (2005) Multilevel least squares approximation of scattered data over binary triangulations, *Comput. Vis. Sci.* **8**, 83–91. doi: 10.1007/s00791-005-0154-7.

Inverse Distance Weighting: https://en.wikipedia.org/wiki/Inverse_distance_weighting.

Jespersen D.C. (1984) Multigrid methods for partial differential equations, in: Golub G.H. (ed), *Studies in numerical analysis*, The Mathematical Association of America **24**, pp. 270–318.

Léger M. (1999) Interpolation from Lagrange to Holberg, in: A. Cohen, C. Rabut, L.L. Schumaker (eds), *Curve and surface fitting*, Vanderbilt University Press, Nashville, TN, pp. 281–290.

Léger M. (2016) *Method of constructing a geological model*. Patent US20160139299.

Lu G., Wong D. (2007) An adaptive inverse-distance weighting spatial interpolation technique, *Comput. Geosci.* **34**, 1044–1055. doi: 10.1016/j.cageo.2007.07.010.

Mitra A., Onyia O., Frangeul J., Parsa A. (2017) Alwyn North Field Ocean Bottom Node (OBN) – A step change in seismic imagery, inversion & interpretation, in: *79th EAGE Conference & Exhibition, Paris*, We A3 14.

Ohtake Y., Belyaev A., Seidel H.-P. (2004) *A multi-scale approach to 3D scattered data approximation with adaptive compactly supported radial basis functions*, Solid Modeling International, pp. 31–39. doi: 10.1109/SMI.2004.1314491.

Perrin M., Rainaud J.-F. (2013) *Shared Earth modeling – knowledge driven solutions for building and managing subsurface 3D geological models*. Technip. ISBN-13: 978-2710810025.

Rogers D.F., Earnshaw R.A. (1991) *State of the art in computer graphics – visualization and modeling*, Springer-Verlag, New York, pp. 225–269.

Sams M., Carter D. (2017) Stuck between a rock and a reflection: A tutorial on low-frequency models for seismic inversion, *Interpretation* **5**, 2, B17–B27. doi: 10.1190/INT-2016-0150.1.

Schneider S. (2002) Pilotage automatique de la construction de modèles géologiques surfaciques, *PhD Thesis*, Université Jean Monnet et Ecole Nationale Supérieure des Mines de Saint-Etienne.

Sobolev Inequality: https://en.wikipedia.org/wiki/Sobolev_inequality.

Tarantola A. (1987) *Inverse problem theory*, Elsevier, Amsterdam.

Appendix

The smoothness of the interpolated surface

It is well known that a finite sum of any $C^{\infty}(R)$ functions give a $C^{\infty}(R)$ function, but this property disappears if the sum becomes infinite. For instance, if $N \to \infty$, the function $S_N(x) = \frac{4}{\pi} \sum_{n=1}^{n=1+2N} \frac{1}{n} \sin\left(\frac{n\pi x}{L}\right)$ tends to a square-function of unit size and length L which is clearly not continuous.

Similarly, what happens to our interpolator when the step size tends to 0? We work with finite regular meshes, and it is possible to define a C^2 polynomial function which is always integrable, whatever the finite step may be. As long as steps are finite there is no problem, but when the step tends to zero we need the Sobolev inequality.

Function $f(i, j)$ belongs to $W^{m,p}(\Omega)$, where m is the number of derivatives in L^P, Ω being an open set of R^n. Sobolev showed (see Ref. "Sobolev Inequality"):

$$k = m - \frac{n}{p} \Rightarrow W^{m,p}(\Omega) \subset C^{k-\varepsilon}('\Omega),$$

where k is the degree of continuity, and ε is a strictly positive real. For our needs, we have $m = 2$ since we minimize the second derivative, $p = 2$ since we use squares in the norm, $n = 2$ since the dimension of a surface is 2, and the result is $k = 1 - \varepsilon$ since the Sobolev inequality. In the following, the result is said *almost* C^1, because of ε. There is a difference between the order of the derivative we minimize, here 2, and the continuity of the result, whatever the mesh, which is here *almost* 1. For applications in geosciences, the continuity of the dip is a consequence of that for surfaces, including the isolated data points.

Note the influence of the dimension, we have simple continuity in 3D for isolated points, not the *almost* continuity of the first derivative in 2D. However, around a differentiable curve with differentiable property, the result will be *almost* C^1 in 3D.

The norm has also an influence. If we choose $p = 1$ instead of $p = 2$, the results are more robust. *When a traveller reaches a fork in the road, the l_1-norm tells him to take either one way or the other, but the l_2-norm instructs him to head off into the bushes.* This citation can be found page 832 in Claerbout and Muir (1973), and also page 303 in Tarantola (1987). The authors argue that the L_1 norm is more robust than the L_2 norm. But the downside is that the result will be only C^0 instead of *almost* C^1 according the Sobolev inequality. The choice of $p = \infty$ leads to the opposite conclusion: the L_∞ norm takes minimum and maximum curvatures into account, which is not at all robust, but according to Sobolev inequality, the result will be *almost* C^2 instead of *almost* C^1. Robustness and smoothness are opposite.