A HAMILTON–JACOBI FRAMEWORK FOR MODELING GEOLOGICAL FOLDING AND DEFORMATION

A HAMILTON–JACOBI FRAMEWORK FOR MODELING GEOLOGICAL FOLDING AND DEFORMATION

Proefschrift

ter verkrijging van de graad van doctor
aan de Technische Universtiteit Delft,
op gezag van de Rector Magnificus prof.ir. K.C.A.M Luyben,
voorzitter van het College van Promoties,
in het openbaar te verdedigen op donderdag 3 juli 2014 om 10:00 uur

door

Øyvind HJELLE

Master of Science from
The University of Trondheim, Norwegian Institute of Technology
geboren te Øvre Årdal, Norway.

Dit proefschrift is goedgekeurd door de promotoren:
Prof.dr.ir. J.T. Fokkema
Prof.dr. S.A. Petersen

Samenstelling promotiecommissie:

Rector Magnificus	voorzitter
Prof.dr.ir. J.T. Fokkema	Delft University of Technology, promotor
Prof.dr. S.A. Petersen	Delft University of Technology / Statoil ASA, promotor
Prof.dr.ir. P.M. van den Berg	Delft University of Technology
Prof.dr. A.M. Bruaset	University of Oslo / Simula Research Lab.
Prof.dr.ir. C.P.A. Wapenaar	Delft University of Technology
Prof.dr. J.D.A.M. van Wees	University of Utrecht / TNO
Prof.dr. X. Cai	University of Oslo / Simula Research Lab.

ISBN 978-82-92593-15-8

Financial support
The research reported in this thesis has been financed by Statoil ASA, and partly by Kalkulo AS, Simula Research Laboratory, and the University of Oslo, Faculty of Mathematics and Natural Sciences.

Preface

In 2004, computer scientists and mathematicians at Simula Research Laboratory joined forces with petroleum geologists and geophysicists at Hydro Oil & Gas[1] and launched an R&D collaboration with the slogan "Making the Invisible Visible". I was very lucky to be invited into the pilot study to pave the road for this long-lasting collaboration. Early in the project we met Steen A. Petersen at Hydro's research department in Bergen. He introduced a set of new concepts for geomodeling to us that he had been working on and programmed as a prototype. I immediately found these ideas very attractive and soon Steen and I were in a fruitful collaboration on software development and later on publication of scientific papers. The software we started developing should later become the Compound Earth Simulator (CES). This collaboration is still lasting. Today, several of my colleagues in Kalkulo[2] are involved in the development of CES, which is now being used on daily basis by geologists and geophysicists in Statoil.

The academic part of the project linked me to TU Delft and Professor Jacob Fokkema who had collaborated with Steen on well log simulation and spatial resolution filtering techniques. In 2010 Steen was appointed Professor at TU Delft, and later I sent a proposal for a PhD to be hosted by TU Delft. Jacob and Steen became promotors for my PhD and it has been a great privilege to work with these visionary scientists. Professor Peter van den Berg at TU Delft was also a good discussion partner during meetings in Delft. I will always have good memories of my visits to beautiful Delft, also of the dinners that Jacob arranged at De Waag where conversations often drifted (away) in philosophical directions when "propositions" were on the agenda.

This PhD would not have been realized without the support from Professor Are Magnus Bruaset. It was Are Magnus who first came up with the idea to

[1] Hydro Oil & Gas merged with Statoil ASA in 2007.

[2] Kalkulo AS is a subsidiary of Simula Research Laboratory.

convert some of my previous research and publications into a PhD, and he has been pushing me constantly forward to finish this work. He has also been a co-author and an acting supervisor at Simula. Hans Petter Langtangen has also been of much help on mathematical issues in this period. Tor Gillberg based his PhD on my early research for Statoil. He gained a huge amount of knowledge on mathematical and numerical aspects of front propagation that I benefited from. Tor was a good collaborator and discussion partner during his four years stay in Kalkulo, until he finished his PhD last year.

Other researchers at Simula have also been of much help: Kent-Andre Mardal, Morten Dæhlen, Aslak Tveito, Bjørn Fredrik Nielsen, Xing Cai and Christian Tarrou (and I should probably have acknowledged more of them). And thanks to my colleagues in the development team of CES who have implemented, tested and refined the results in this thesis to professional software.

Finally, I thank my daughters Ingvild and Guro, and my wife Marit for their support and their patience during these years. Thanks to Guro for preparing the cover of this book, which is based on photos of beautiful fold structures in a road cut in Bergen.

Øyvind Hjelle
Delft, April 2014

Contents

Chapter 1

Introduction and Background

When investigating the geological evolution of a region, concurrent theories will arise and compete as geologists try to fit together information provided by collected data. Traditionally, geology has been centered on a qualitative understanding of physical processes spanning tens and hundreds of million years. However, through innovative use of different data sources, such as seismic profiles, electromagnetic recordings, and bore samples, computer-based models have become quantitative supplements and corrections to the qualitative interpretation. Such models are of particular value to oil and gas companies searching for hydrocarbon reserves across wide ranges of geological settings, often of very high complexity. For example, computer-based models of geological folding and faulting are important when mapping out paleogeographical features, such as when identifying geological structures that have supported accumulation of sediments in a basin.

Over the past decades, advances in computer power and memory have paved the way for new and more advanced software for subsurface modeling. Interpretation systems managing huge seismic volumes in three dimensions are now commonly integrated with a range of other software tools and workflows such as structural model building, property modeling, velocity modeling, time-to-depth conversion, well planning, restoration, and more. Examples of such software platforms are Petrel (Schlumberger), RMS (Roxar/Emerson) and SKUA (Paradigm).

Statoil ASA's Compound Earth Simulator (CES) is another geomodeling system taking advantage of the rapid growth in computer power and memory. CES is founded on a set of new ideas and a new concept for geomodeling,

referred to as Earth Recursion (Petersen and Hjelle, 2008), which will be discussed below. CES aims to make geomodeling truly four-dimensional, in that the geological evolution as a whole is reflected in the model, instead of being limited to the present day geology. The geological evolution is described as a sequence of processes (possibly overlapping) acting over time, starting with deposition of sediments. Deformations caused by tectonic changes such as folding and faulting are superimposed on the model, new depositions are imposed, and erosion, compaction and other structural changes are included as they take place over geological time. Along with these events and structural changes, the geological volume is populated with physical properties such as velocity, density, porosity, gamma radiation, saturation, etc. One key concept in CES is how the spatial distribution of properties is controlled by surfaces representing horizons, faults, fluid contacts etc., and curves representing well traces. Let p_j denote a one-dimensional property function that depends on the distance from a point $\mathbf{x}$ in 3D space to one or more structural elements, and let $d_i(\mathbf{x})$ be the distance from $\mathbf{x}$ to the ith structural element. Conceptually, a property value P_j at $\mathbf{x}$ is then given by

$$P_j(\mathbf{x}) = C_{\texttt{geo-rules}}(p_j(d_i(\mathbf{x}),\, i = 1, \ldots, N)). \tag{1.1}$$

Here, the functional $C_{\texttt{geo-rules}}$ represents geological rules that specify how contributions from different geometries are combined and composed into a final property value at $\mathbf{x}$. $C_{\texttt{geo-rules}}$ can be quite complex and may require an algorithmic representation, but conceptually this equation is simple since it only depends on one-dimensional property functions and distance fields. Complex and realistic property distributions can be established by combining distance fields. For example, when describing saturation changes along a well-bore, the distance field of the original fluid contact can be used, and then the relevant property can be modified with the help of a property derived from the distance field of the well geometry. (See examples in Petersen *et al.* (2007) both in two and three dimensions.) The starting point for the research in this dissertation was to derive distance fields reflecting the different folding classes defined in the classical literature of structural geology by Ramsay (1967) and Hudleston (1973). This is covered in Chapters 2 and 3 in this thesis and have been published in Hjelle and Petersen (2011), and in Hjelle *et al.* (2013). The results are used directly in the property distribution function above to determine $d_i(\mathbf{x})$.

The concept of Earth Recursion (Petersen and Hjelle, 2008) formulates the geological evolution as recurrently acting geological processes. By this concept, spatial and temporal distribution of properties can be regarded as a recursive sequence that can describe highly complex property distributions that reflect the interference of any number of geological processes, such as onlapping, faulting, folding and compression. Earth Recursion forms the basis of

a set of concepts to describe and combine a process-based data restoration (dR) workflow and a model reconstruction (mR) workflow for simultaneous seismic interpretation and model building (Petersen *et al.*, 2012). In the dR-workflow, a depth seismic section is gradually altered by reverse, mainly structural, processes in order to restore the seismic section as if it were recorded immediately after the time of deposition (Fig. 1.1, left column). Thus, the impact of faults, folds, compaction etc. is gradually removed by invoking the inverse of these processes. The section looks "younger" and stratigraphic interfaces appear easier to interpret. The mR-workflow (Fig. 1.1, right column) starts with distribution of properties as it would appear immediately after the time of deposition. Then the normal version of the previous reverse geological processes in the dR-workflow are applied, now ordered forward in geological time. The final property distribution is a reconstruction of the present day distribution. The successive changes in both sequences of the dR/mR-workflows act recurrently by the concept of Earth Recursion described above. A more detailed description of the workflows in the example depicted in Fig. 1.1 can be found in Petersen *et al.* (2012). Chapter 4 in this thesis presents a mathematical and numerical framework for modeling processes in the dR/mR-workflows.

Fig. 1.2 shows a 2D example from Petersen *et al.* (2012) of the dR/mR-workflows implemented in CES. The workflows are applied to a real case from the North Sea starting with a seismic section (dR1). The original depositional system (dRn) was reached after approximately 20 recurrently acting processes initiated from a set of interpreted horizons and faults (lower right). The mR-workflow (middle row) was initiated with properties distributed in a sheet-like internal layering, including some pinch-outs (mR1). Then the layers were transferred back to their present position (mRn) with some additional processes added (truncation, overburden onlap and deformation). In the last step, a synthetic section (lower left) was derived from mRn by a spatial resolution filtering technique (Toxopeus *et al.*, 2008). This section is then compared to the original seismic in dR1 to verify correctness of interpretations and predicted processes included in the dR-workflow. Note that any instance with property distribution in the mR-sequence has its counterpart with restored seismic data in the dR-sequence. Thus, comparison between predicted and observed data can take place at any time in the geological history. This can be used to check underlying hypotheses and to experiment with different scenarios along the geological timeline. (See Petersen *et al.* (2012) for more details.)

In CES, geological models are represented as continua defined on regular grids. Both property fields and distance fields, and derived fields use this simple format. This also serves as a computational grid for finite difference solvers to compute distance fields (Chapter 3), and to compute scalar fields for representing a parameterization of the 3D space (Chapter 4). Deformations representing

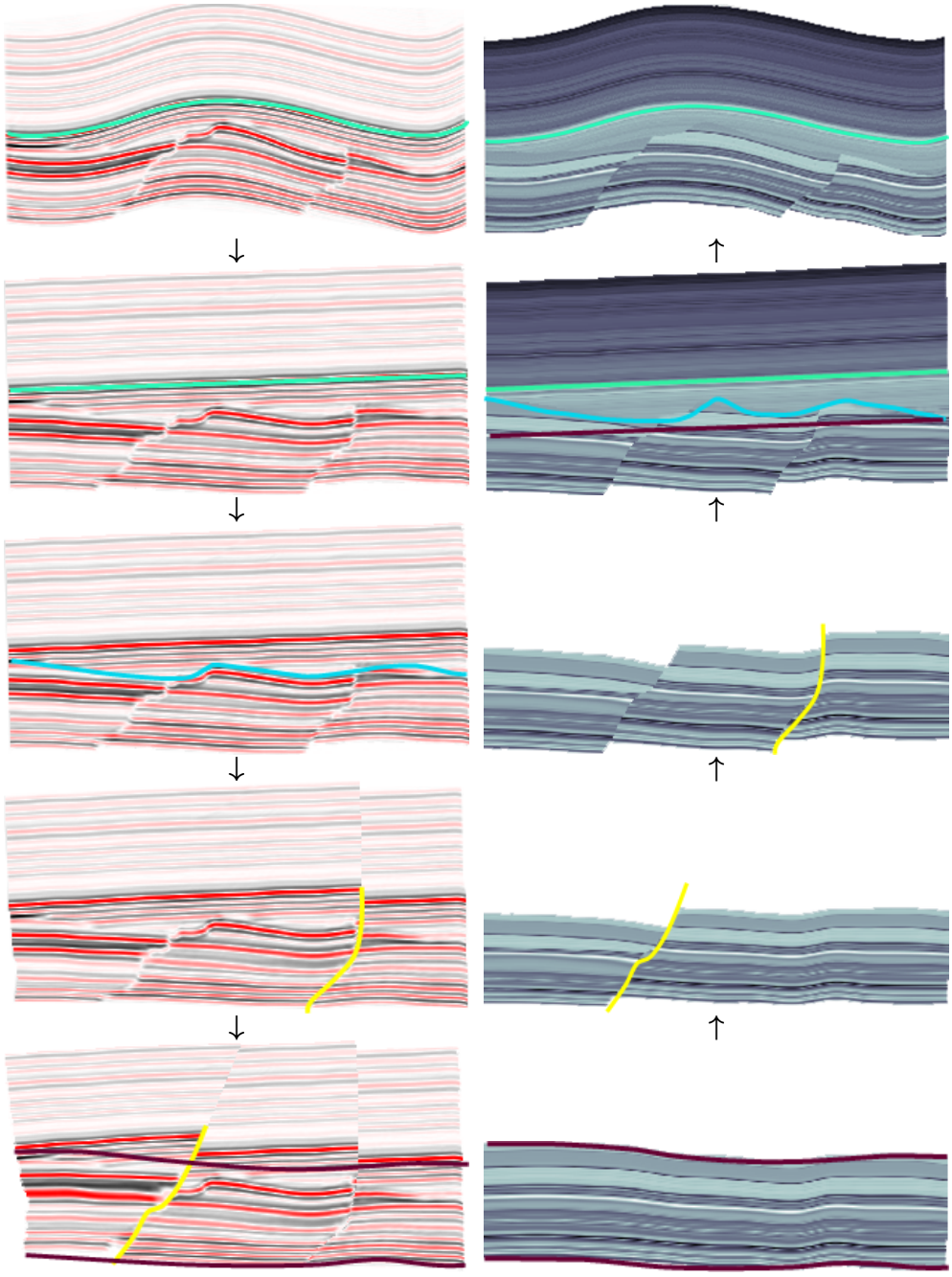


Fig. 1.1: Synthetic example of the dR/mR-workflows. Left column: dR-workflow where a seismic section is altered by inverse geological processes. Right column: mR-workflow with the normal version of the same geological processes. The arrows indicate the direction of the workflows. (Modified from Petersen *et al.* (2012).)

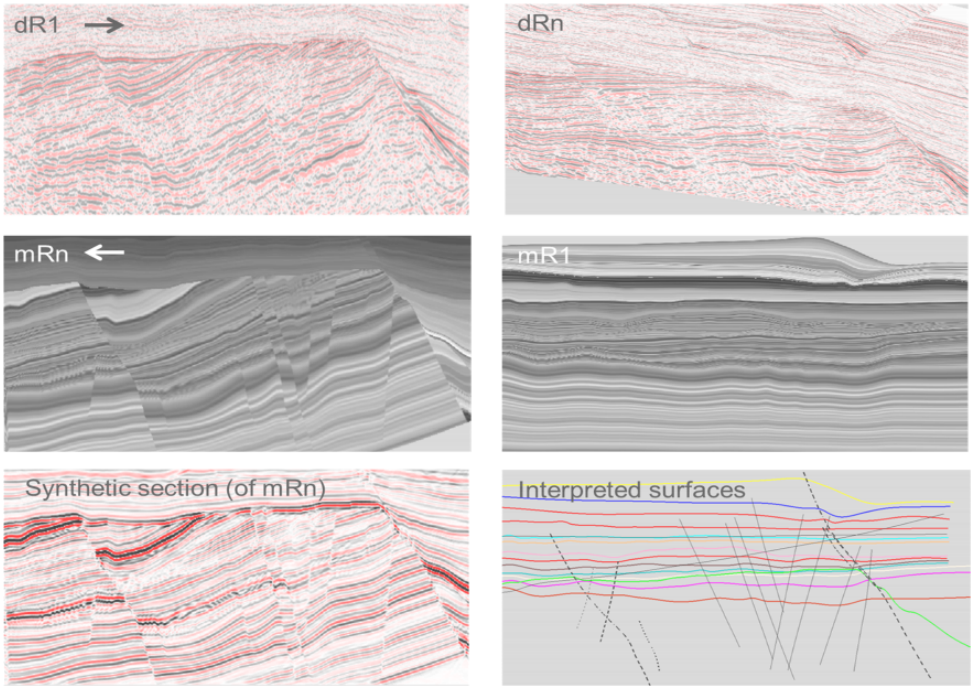


Fig. 1.2: Real case from the North Sea of the dR/mR-workflows. Upper row: dR-workflow with a seismic section (dR1) and original depositional system (dRn) reached after approximately 20 processes. Middle row: mR-workflow, reversing all the processes above, starting with mR1 and resulting in mRn. Lower row: A synthetic seismic section based on mRn, and interpreted faults and horizons used to initiate processes. (From Petersen *et al.* (2012).)

restoration in the dR-workflow, or reconstruction in the mR-workflow, are also computed and represented on the same grid. Curves, surfaces and other geometric objects are only used to control space when computing distance fields and distributing properties initially. Their geometric representations are always kept separated from the computational model, however, these geometries are implicitly defined as isocontours and isosurfaces of their respective distance fields. This is in contrast to traditional geomodeling with a structural representation where interpreted surfaces representing horizons, faults, unconformities, etc. are combined to form a structural model, and where a tetrahedral mesh is constructed to honor the surfaces of the structural model.

This meshless volumetric approach may cause some operations to be memory consuming and computational demanding when the model is large. For example, computation of distance fields on large grids in three dimensions is

challenging with traditional methods. There is ongoing research with promising results to compute distance fields on parallel multicore CPU and GPU architectures. This reduces computing times from several minutes to seconds, thus enabling interactive restoration and reconstruction of huge complex geological models with high-resolution in three dimensions (Gillberg, 2013; Gillberg *et al.*, 2014).

Note that the property function (1.1) only enters the spatial dimension through the distance functions $d_i(\mathbf{x})$. This observation simplifies the implementation of the numerical kernel in CES, which is done generically, more or less independently of the spatial dimension. Another important design feature is that the spatial dependency of a property value is relative to one or more structural elements in the model. Therefore, properties can stay fixed within a geological time window in their relative position as structural changes or deformation of the volume take place, while the same properties vary when observed in the Euclidean space. However, properties can be overprinted or changed, for example through chemical processes such as diagenesis. As shown in Chapter 4, this principle simplifies the implementation of operations along the geological time axis, for example processes for restoration by de-faulting and unfolding horizons. Moreover, the geological rules in $C_{\texttt{geo-rules}}$ are designed to keep the number of independent degrees of freedom as low as possible. When structural information is altered or property functions are perturbed, or even geological rules are changed, the updated information propagates automatically in space and time to the whole model. This makes the model highly editable and makes it possible to experiment with a number of different geological hypotheses and scenarios along the geological timeline in the dR/mR-workflows.

The research reported in this thesis was motivated by the concept of Earth Recursion and the process-based data restoration (dR) and model reconstruction (mR) workflows outlined above. The objective of the research has been to develop a mathematical and numerical framework that can lay the foundation for implementation of processes in the proposed dR/mR-workflows. The results in the proceeding chapters have been realized as software libraries and implemented in CES, and have been used by engineers and geoscientists in Statoil ASA in a number of industrial cases. It should be emphasized that the results have wide applicability beyond the use in CES, and most of the material in this thesis has been published in international journals. Results from Chapters 2 and 3 have also been the starting point for other recent academic work in Gillberg *et al.* (2012), and the research on multicore CPU and GPU architectures mentioned above (Gillberg, 2013; Gillberg *et al.*, 2014).

The main results that form the basis of this dissertation are reported in Chapters 2, 3, 4, and Appendix A. Chapters 2, 3 and 4 comprise a coherent

presentation of the mathematical and numerical frameworks for the concepts outlined above in Chapters 2 and 3, respectively, and extensions and applications of these frameworks in Chapter 4. Appendix A presents results that are are important for realization of the results from the other chapters as software and implementation in practical applications. More specifically:

Chapter 2 presents a unified mathematical framework for representing fold classes defined in the classical literature of structural geology by Ramsay (1967), Hudleston (1973), and others. A static Hamilton-Jacobi equation is derived for modeling folds as generalized distance fields. Metric properties such as gradient (dip and strike), curvature, and their spatial variations are also derived. (Based on the journal paper Hjelle and Petersen (2011))

Chapter 3 presents the numerical counterpart to the mathematical model derived in Chapter 2. A numerical scheme is derived to solve the Hamilton-Jacobi equation by upwind finite differences, and represent folds as continua on grids covering the geological volume. Numerical examples from CES are given, where the fast marching method is used to generate the distance fields. (Based on the journal paper Hjelle *et al.* (2013).)

Chapter 4 extends the mathematical and numerical frameworks in the two preceding chapters and applies the results to data restoration and model reconstruction with numerical examples from the dR/mR-workflows implemented in CES. A novel method for parameterization of the 3D space is derived for this purpose and given a meshless representation on the computational grid. This parameterization is also used to derive metric tensors for quantifying strain in connection to restoration.

Chapter 5 summarizes and concludes from the preceding chapters, and gives some directions of further research.

Appendix A presents a multilevel approximation scheme for generating smooth surfaces from huge scattered data sets. The method is well suited to handle interpreted horizon and fault data extracted from seismic volumes, for example by autotracking, and generates surfaces as input to distance field computations (Chapter 3). (Based on the journal paper Hjelle and Dæhlen (2005), and Chapter 8 in the book Hjelle and Dæhlen (2006).)

Appendix B presents Newton iteration schemes for computing the closest point to parametric curves and surfaces, and Newton iteration schemes for intersection of parametric curves and surfaces with straight lines. These operators are used in the initialization step of the fast marching algorithm presented in Chapter 3.

Appendix C contains finite difference operators for numerical derivatives used in front propagation algorithms, and numerical operators for calculating curvature.

Chapter 2

A Hamilton-Jacobi Framework for Modeling Folds in Structural Geology[1]

A novel mathematical framework for modeling folds in structural geology is presented. All the main fold classes from the classical literature: parallel folds, similar folds, and other fold types with convergent and divergent dip isogons are modeled in 3D space by linear and non-linear first-order partial differential equations. The equations are derived from a static Hamilton-Jacobi equation in the context of isotropic and anisotropic front propagation. The proposed Hamilton-Jacobi framework represents folded geological volumes in an Eulerian context as a time of arrival field relative to a reference layer. Metric properties such as distances, gradients (dip and strike), curvature, and their spatial variations can then be easily derived and represented as 3D continua covering the whole geological volume. In this chapter we intend to quantify the shapes of folds, and not to model the physics of fold formation, although strain states can also be analyzed by an extension of the model as presented in Chapter 4.

Classical methods for fold classification in structural geology focus on practical methods in the sense that geologists can use these methods in the field to recognize different fold types and describe folds quickly. Classification has mostly been based on the shape of individual surfaces and layers of folds and how two surfaces that enclose a layer interrelate. Ramsay (1967) classifies different fold styles based on layer thickness variation (layer-thickness-to-dip-

[1]This chapter is based on Hjelle and Petersen (2011)

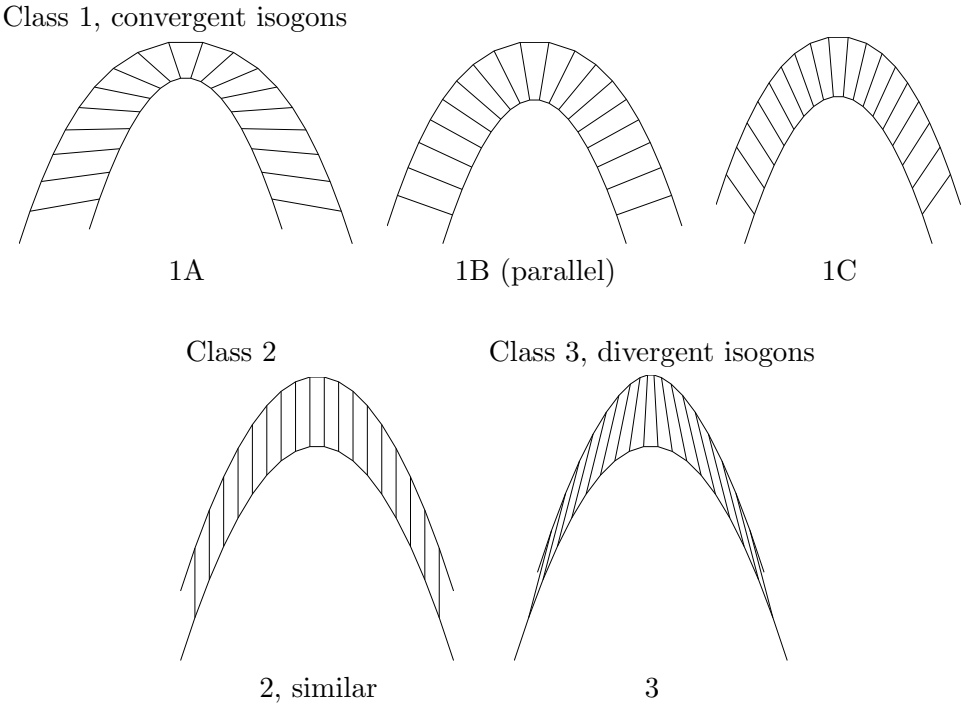


Fig. 2.1: The five fold classes defined in Ramsay (1967), modeled in the Hamilton-Jacobi framework in sections 2.3.1–2.3.5. The orientations of the fold profiles are standardized as upright anti-forms, concave downward, with axial direction pointing upward. Dip isogons are shown between the two layer boundaries for all fold classes. (Modified from Ramsay (1967, p. 365).)

angle-ratio) along the limb of a fold in a 2D profile section. A related method by the same author is based on dip isogon characteristics of the upper and lower boundaries of a layer, and how these boundaries interrelate by dip isogons across the layer. Yet another method, by Hudleston (1973), is based on the ratio between the dip isogon angle and the limb dip. These three classification methods lead to the same division into five main fold classes that occur commonly in naturally deformed rocks: Class 1A, Class 1B, Class 1C, Class 2, and Class 3 in the literature. The different fold classes are shown in Fig. 2.1 and will be explained in detail below in view of the three classification methods.

Other methods for classifying folds are also proposed in the literature. A method given by Srivastava and Lisle (2004) models the profile sections of folded surfaces using cubic Bézier curves (Farin, 2002). Piecewise cubic polynomial functions approximate the shape of a fold limb between the hinge

and the inflection point. A classification into "shape groups" is made based on two parameters related to the shape of the resulting Bézier curve: a parameter related to the distribution of curvature along the limb, and one related to the ratio of amplitude to wavelength. This classification is carried out for one limb and one layer boundary at a time and provides no information on how the surfaces are related to each other. Stabler (1968) and Hudleston (1973) use simple Fourier analysis and trigonometric functions to represent the geometry of a layer, as opposed to polynomial functions used by Bézier curves in the method mentioned above.

The aim of the research presented here is to establish a new mathematical framework for modeling folds and to relate this framework directly to the classical methods mentioned above that divide folds into the five fold classes. Before we derive the mathematical model, we first explain in more detail the classification methods based on layer thickness variation and dip isogons along 2D profile sections in Sect. 2.1. Then Sect. 2.2 provides some mathematical foundation on front propagation and the static Hamilton-Jacobi equation. The new mathematical framework for modeling folds in both 2D and 3D is described in detail in Sect. 2.3.

2.1 Classical Fold Classification

2.1.1 Classification by Layer Thickness Variations

Thickness variation along the limb of a fold can be represented by the ratio $t'_\alpha = t_\alpha/t_0$, where t_α is the orthogonal layer thickness at dip angle α and t_0 is the thickness at the hinge (Fig. 2.2(a)) Following common conventions in the literature, we assume that the orientation of the fold profile is standardized as an upright antiform, concave downward, where the dip angle $\alpha > 0$ on the right limb, $\alpha < 0$ on the left limb, and $\alpha = 0$ at the hinge. By plotting t'_α against the dip angle as in Fig. 2.2(b), division into the different fold classes shown in Fig. 2.1 is carried out as follows.

The Class 1B fold, commonly called the parallel fold, has a constant t'_α-ratio equal to one, that is, the thickness measured orthogonally across the layer boundaries of a parallel fold is constant.

The Class 2 fold, commonly called the similar fold, has $t'_\alpha = \cos\alpha$. Thus, the thickness of a similar fold measured orthogonally to the layer boundaries varies by the thinning of the limbs relative to that of the hinge zone, but the thickness in the direction parallel to the axial surface is constant.

The Class 1C and Class 3 folds also have thinning fold limbs with, respectively, less and more thinning than that of the similar fold. Class 1A has thickening of the fold limbs relative to that of the hinge zone.

Thickness variation along the limb of a fold can also be based on axial plane thickness. This also leads to the fold classes described above (Ragan, 2009, p. 385).

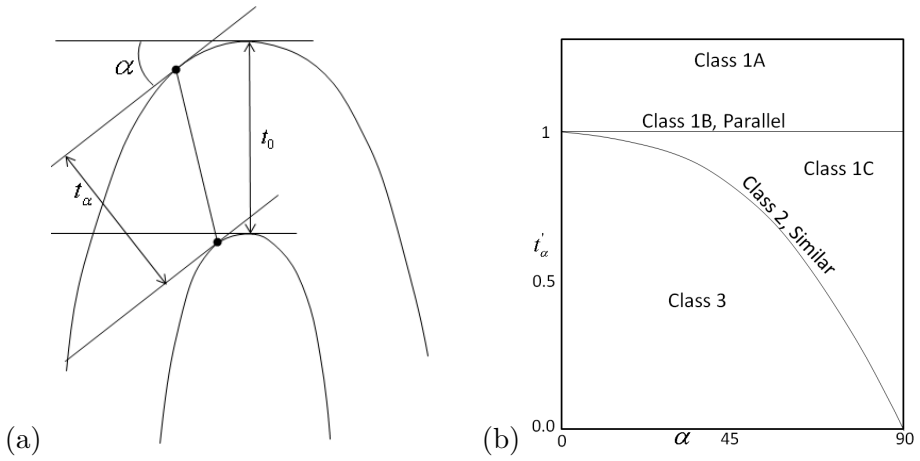


Fig. 2.2: (a): Thickness variation along the limb of a Class 1C fold. The line with bullets is a dip isogon. (b): Orthogonal layer thickness ratio t'_α against the dip angle α for the different fold classes. (Modified from Ramsay and Huber (1987, p. 348 and 349).)

2.1.2 Classification by Dip Isogons

In the context of the mathematical model we derive below, classification with dip isogons is of particular interest, since dip isogons coincide with the characteristic curves of the static Hamilton-Jacobi equations we use to model the different fold classes. A dip isogon connects two points with equal dip angles on the upper and lower boundaries of a layer (Fig. 2.1). Class 1A-C folds have convergent dip isogons, as seen when traced from the outer toward the inner arc of a fold. Class 1B folds (parallel) have dip isogons that are orthogonal to the layer boundaries. Class 1A and Class 1C folds have dip isogons that are, respectively, strongly convergent and weakly convergent compared to those of Class 1B. Class 2 folds (similar) have parallel dip isogons, and Class 3 folds have divergent dip isogons, as seen when traced from the outer towards the inner arc. The second and fourth columns in Table 2.1 summarize how the fold classes are characterized by layer thickness variations and dip isogons. Another

Fold class	t'_α	ϕ ($\alpha > 0$)	Dip isogons
Class 1A	> 1	< 0	strongly convergent
Class 1B	$= 1$	$= 0$	orthogonal to boundaries
Class 1C	$\cos\alpha < t'_\alpha < 1$	$0 < \phi < \alpha$	weakly convergent
Class 2	$= \cos\alpha$	$= \alpha$	parallel
Class 3	$< \cos\alpha$	$> \alpha$	divergent

Table 2.1: Characterization of the fold classes by layer thickness variation t'_α, dip isogon angle ϕ, and convergence, divergence and parallelism of dip isogons.

complementary classification method by Hudleston (1973) is based on plotting the dip isogon angle ϕ, or isogon angle for short, against the dip angle α. The isogon angle is measured relative to the normal vector of a layer boundary such that $\phi = 0$ for Class 1B folds, and it is positive when measured counterclockwise from the normal vector. At the hinge, $\phi = 0$ for all fold classes. The third column in Table 2.1 shows how the isogon angle varies for the different fold classes on the right limb and how the isogon angle relates to the dip angle. See also Ragan (2009, p. 387), for more details and how this classification method can be used to analyze cleavage orientation around folded layers.

Other useful geometric characteristics of folds follow directly from the classifications above. For example, in Class 3 folds the curvature of the inner boundary is less than that of the outer boundary. In Class 2 folds the curvatures of the inner and outer boundaries are the same, while for Class 1 folds the curvature of the inner boundary is greatest and decreases along dip isogons toward the outer boundary. We also note that for Class 1B and Class 2 folds the dip isogons between two layers are of equal length.

2.2 Mathematical Foundation

The Hamilton-Jacobi framework we present in Sect. 2.3 is an Eulerian approach, which represents fold geometry as continua in the whole 3D volume of geological layers. The mathematical model is founded on a front propagation analogy by which a continuum of layer boundaries of a fold is modeled as a propagating front starting from a reference layer boundary. The propagating front evolves in space and time, and we seek its unique first arrival time value everywhere in the domain of interest. The isosurfaces (isocurves in 2D) of this time of arrival field is then associated with a continuum of layer boundaries. The notion of time in this context can be associated with geological

time, although a direct correspondence is not immediately present. We will return to this later. The front propagation analogy is stated as a boundary value problem with partial differential equations (PDEs) derived from the static Hamilton-Jacobi equation. This approach to modeling folds is different from classical methods, which have mostly been based on a geometric description of the boundary surfaces of folds.

To produce the numerical examples presented throughout this chapter, we developed a variant of the fast marching method for anisotropic problems which solves the PDEs on a finite difference grid (Chapter 3). The fast marching method was first introduced by Sethian (1996) for isotropic front propagation, and later extended to anisotropic front propagation by Vladimirsky (2001), and Sethian and Vladimirsky (2003).

2.2.1 Front Propagation and Time of Arrival Fields

Let Γ be a curve evolving in 2D, or a surface evolving in 3D, and assume that every point on Γ moves in the direction normal to Γ, governed by a speed function $F(\mathbf{x}, \mathbf{n})$, where $\mathbf{n}$ is the unit normal vector to Γ as it passes through the point $\mathbf{x}$ (Fig. 2.3). In the following we call Γ a propagating front and the

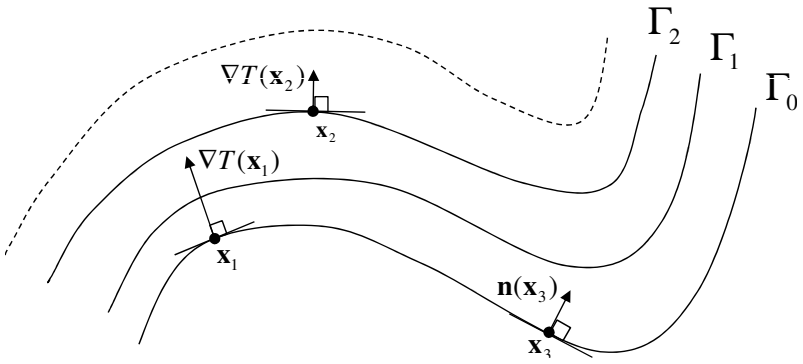


Fig. 2.3: A front moving with variable speed $F(\mathbf{x}, \mathbf{n})$ in its normal direction from the initial front Γ_0.

motion of Γ front propagation. We assume that F is strictly positive such that the front can pass through a point only once. Furthermore, let $T(\mathbf{x})$ represent the time of arrival (or traveltime) of Γ at $\mathbf{x}$ starting from the initial front Γ_0 at time $t = 0$. Then the level sets of T correspond to the evolving front at different times t,

$$\Gamma_t = \{\mathbf{x} \in R^{d\in\{2,3\}} \mid T(\mathbf{x}) = t\}. \tag{2.1}$$

The normal vector $\mathbf{n}(\mathbf{x})$ has the same direction as the gradient of T in $\mathbf{x}$. In three dimensions with $\mathbf{x} = (x, y, z)$, the gradient is $\nabla T(\mathbf{x}) = (\partial T/\partial x, \partial T/\partial y, \partial T/\partial z)$. Then the normal can be expressed as

$$\mathbf{n}(\mathbf{x}) = \frac{\nabla T(\mathbf{x})}{\|\nabla T(\mathbf{x})\|},$$

where $\|\cdot\|$ denotes the Euclidean norm. We observe that ∇T at a point $\mathbf{x}$ is orthogonal to the level set passing through $\mathbf{x}$. Here we assume that Γ_t and thus T are sufficiently smooth such that the gradient and the normal vector are well defined for all $\mathbf{x}$. But this is not necessarily true everywhere, since the propagating front can develop cusps and corners where the gradient is not well defined. We return to this in Sect. 2.3.1.

Consider a point $\mathbf{x}$ on the front that is moving with speed $F(\mathbf{x}, \mathbf{n})$ in the normal direction of Γ. Let the point move a small distance δ to a new point $\bar{\mathbf{x}} = \mathbf{x} + \delta\mathbf{n}(\mathbf{x})$. Using the fact that distance=speed×time, we have

$$\frac{T(\bar{\mathbf{x}}) - T(\mathbf{x})}{\delta} \approx 1/F(\mathbf{x}, \mathbf{n}),$$

and in the limit as $\delta \to 0$ we obtain the boundary value problem

$$\begin{aligned} &\|\nabla T(\mathbf{x})\| = 1/F\left(\mathbf{x}, \tfrac{\nabla T(\mathbf{x})}{\|\nabla T(\mathbf{x})\|}\right), \\ &\text{given } \; T = 0 \text{ on } \Gamma_0. \end{aligned} \tag{2.2}$$

This is a non-linear first-order PDE that belongs to the class of static Hamilton-Jacobi equations or more generally, to hyperbolic conservation laws (LeVeque, 1992). The boundary value problem will serve as a unified model to model the different fold classes defined in Sect. 2.1. We will frequently refer to

$$H(\mathbf{x}, \nabla T) = \|\nabla T\| F\left(\mathbf{x}, \frac{\nabla T}{\|\nabla T\|}\right) \tag{2.3}$$

as the Hamiltonian such that a more general form of the static Hamilton-Jacobi equation can be written $H(\mathbf{x}, \nabla T) = 1$. Since F depends on $\mathbf{n}$ in general, the front propagation is anisotropic. Suppose, on the other hand, that the front propagation is isotropic, that is, $F(\mathbf{x}, \mathbf{n}) = F(\mathbf{x})$. Then Eq. (2.2) reduces to the eikonal equation

$$\|\nabla T(\mathbf{x})\| = 1/F(\mathbf{x}). \tag{2.4}$$

In the special case with constant speed $F = 1$, T corresponds to a distance field relative to Γ_0 where the distance is zero.

2.2.2 Characteristics

The characteristic curves of a PDE, or just characteristics, are curves in the solution domain along which the PDE can be reduced to an ordinary differential equation (ODE). Then the solution value $T(\mathbf{x})$ in a point $\mathbf{x}$ depends only on other values of T along the characteristic passing through $\mathbf{x}$. Let the static anisotropic Hamilton-Jacobi equation in the boundary value problem of Eq. (2.2) be expressed by the Hamiltonian as

$$H(\mathbf{x}, \mathbf{p}) = 1,$$

where $\mathbf{p} = \nabla T(\mathbf{x})$. Let $(\mathbf{x}(t), \mathbf{p}(t))$ represent curves in the solution domain. Along these curves we have

$$\frac{dH(\mathbf{x}(t), \mathbf{p}(t))}{dt} = \nabla_{\mathbf{x}} H \cdot \frac{d\mathbf{x}}{dt} + \nabla_{\mathbf{p}} H \cdot \frac{d\mathbf{p}}{dt} = 0, \tag{2.5}$$

where $\nabla_{\mathbf{x}}$ and $\nabla_{\mathbf{p}}$ are gradient operators with respect to $\mathbf{x}$ and $\mathbf{p}$, respectively[1]. Then, $d\mathbf{x}/dt$ and $d\mathbf{p}/dt$ can be considered as a system of (coupled) ODEs. In particular, $d\mathbf{x}/dt$ represent tangent vectors of $(\mathbf{x}(t), \mathbf{p}(t))$. The following system of ODEs satisfies Eq. (2.5).

$$\frac{d\mathbf{x}}{dt} = \nabla_{\mathbf{p}} H, \tag{2.6}$$

$$\frac{d\mathbf{p}}{dt} = -\nabla_{\mathbf{x}} H. \tag{2.7}$$

We return to these equations in Sect. 2.3, where we model the different fold classes by different Hamiltonians and thus by different characteristic curves. In particular, we will see that there is an important link between characteristics and dip isogons.

The static Hamilton-Jacobi equation and the boundary value problem of Eq. (2.2) arise naturally in geometric optics for wave propagation. Geometric optics is a simplified model to describe how light propagates as geometric rays, and has many applications in partial differential equations. In particular, the characteristics defined by Eq. (2.6) and (2.7) play an important role in this field. We sometimes use terminology established in geometric optics, although our focus is on the description of folds in structural geology. The following establishes some links and adopts some notations.

In geometric optics, a level set Γ_t of T given by Eq. (2.1) is called a wavefront and $\mathbf{n}(\mathbf{x}) = \nabla T(\mathbf{x})/\|\nabla T(\mathbf{x})\|$ is the wavefront normal in $\mathbf{x}$. The

[1]Throughout this thesis, when the gradient operator ∇ is without a subscript, the gradient is with respect to the spatial variables $\mathbf{x} = (x, y, z)$ in 3D and $\mathbf{x} = (x, y)$ in 2D.

gradient $\mathbf{p}$ is sometimes called a slowness vector, since the elements of $\mathbf{p}$ are the reciprocals of velocity. Then

$$V_p(\mathbf{x}, \mathbf{n}(\mathbf{x})) = \frac{1}{\|\mathbf{p}(\mathbf{x})\|}$$

is called the phase speed (in the normal direction of the wavefront), and we get

$$\mathbf{p}(\mathbf{x}) = \frac{\mathbf{n}(\mathbf{x})}{V_p(\mathbf{x}, \mathbf{n}(\mathbf{x}))}.$$

With the Hamiltonian on the form $H(\mathbf{x}, \mathbf{p}) = 1$, the phase speed can be written

$$V_p(\mathbf{x}, \mathbf{n}(\mathbf{x})) = \frac{H\left(\mathbf{x}, \mathbf{p}(\mathbf{x})\right)}{\|\mathbf{p}(\mathbf{x})\|} = H(\mathbf{x}, \mathbf{n}(\mathbf{x})). \tag{2.8}$$

In the last equality we have moved the denominator inside the parenthesis assuming that H is homogeneous of degree 1 in the second argument. The characteristic curves $(\mathbf{x}(t), \mathbf{p}(t))$ are called ray trajectories. Eq. (2.6) defines the group velocity vector, which points in the ray direction. The magnitude of the group velocity vector is the group speed,

$$v_g(\mathbf{x}, \mathbf{p}) = \left\| \frac{d\mathbf{x}}{dt} \right\| = \|\nabla_{\mathbf{p}} H\|.$$

In general, the group speed depends on both position and the slowness vector (traveltime gradient), implying so-called directional dependence.

In a homogeneous anisotropic medium, the phase speed depends only on the direction of the front, $V_p(\mathbf{x}, \mathbf{n}(\mathbf{x})) = V_p(\mathbf{n}(\mathbf{x}))$, and the Hamiltonian becomes $H(\mathbf{x}, \mathbf{p}) = H(\mathbf{p})$. In an isotropic medium, the group velocity vector $d\mathbf{x}/dt$ and the slowness vector $\mathbf{p}$ have the same direction, and group speed equals phase speed. In an anisotropic medium this is not the case. Numerical schemes and algorithms for solving the boundary value problem of Eq. (2.2) must take this difference into account. The isotropic problem can be solved numerically on a regular grid directly and elegantly with Sethian's fast marching method (Sethian, 1999b), but the anisotropic problem requires other numerical schemes.

By using the results above, we can add a third ODE to Eq. (2.6) and (2.7), for the time of arrival T along the characteristic curves. Since $T = T(\mathbf{x})$ we have

$$\frac{dT(\mathbf{x})}{dt} = \nabla T \cdot \frac{d\mathbf{x}}{dt} = \mathbf{p} \cdot \nabla_{\mathbf{p}} H(\mathbf{x}, \mathbf{p}),$$

and from Eq. (2.8) we have

$$H(\mathbf{x}, \mathbf{p}) = H(\mathbf{x}, \mathbf{n}) \, \|\mathbf{p}\|.$$

Then the group velocity vector can be written

$$\nabla_{\mathbf{p}} H(\mathbf{x}, \mathbf{p}) = H(\mathbf{x}, \mathbf{n}) \frac{\mathbf{p}}{\|\mathbf{p}\|} = V_p(\mathbf{x}, \mathbf{n}) \frac{\mathbf{p}}{\|\mathbf{p}\|} = \frac{\mathbf{p}}{\|\mathbf{p}\|^2},$$

which leads to

$$\frac{dT}{dt} = \mathbf{p} \cdot \nabla_{\mathbf{p}} H = 1. \tag{2.9}$$

We also mention the method of characteristics, which is a technique to solve more general hyperbolic PDEs: First the characteristic curves are found and the ODEs along the characteristics are established. Once the coupled system of ODEs (corresponding to Eq. (2.6) and (2.7)) is solved, solution values of the original PDE along the trajectories are found (here by solving Eq. (2.9)).

2.3 The Mathematical Model from an Eulerian Perspective

This section specializes the mathematical model from Sect. 2.2 to represent the different fold classes described in Sect. 2.1. The model is founded directly on the front propagation analogy by which we regard a continuum of layer boundaries of a fold as a propagating front starting from an initial layer boundary Γ_0. The boundary value formulation in Eq. (2.2) is used to model this motion of the front such that level sets Γ_t given by Eq. (2.1) correspond to the continuum of layer boundaries of the fold. In general, we use the anisotropic formulation where the motion is governed by a speed function $F(\mathbf{x}, \mathbf{n})$ that depends both on the position $\mathbf{x}$ and the direction $\mathbf{n} = \nabla T / \|\nabla T\|$ of the front. Typically, Γ_0 corresponds to the lower (older) boundary and the level sets Γ_t correspond to boundaries of younger layers deposited at a later geological time. Note that the variable t does not necessarily correspond to geological time here. The boundary problem of Eq. (2.2) with unknown T is typically solved numerically on a regular or triangular grid. Metric properties such as distances, gradients, and curvature can then also be easily derived and represented by scalar fields and vector fields covering the whole geological volume. A key feature of the mathematical model is that the characteristics of the Hamilton-Jacobi equations representing the different fold classes coincide with the dip isogons used in the fold classification. This has important theoretical and practical implications.

The following designs appropriate speed functions $F(\mathbf{x}, \mathbf{n})$ to the front propagation model that result in time of arrival fields which correspond to the different fold classes defined in Sect. 2.1. Each of the five fold classes are treated separately through Sect. 2.3.1 (Class 1B), Sect. 2.3.2 (Class 2),

Sect. 2.3.3 (Class 1C), Sect. 2.3.4 (Class 1A), and Sect. 2.3.5 (Class 3). Then we summarize with a unified mathematical framework for representing all the fold classes in Sect. 2.4.

2.3.1 Class 1B Fold (Parallel)

A class 1B fold has no layer thickness variation in its normal direction along the boundary of a layer. In the context of front propagation, this corresponds to a speed model with constant speed in the outward normal direction of a level set Γ_t. That is, the speed does not depend on the orientation of the front, and so the front propagation is isotropic. Then the static Hamilton-Jacobi equation (2.2) reduces to the eikonal equation

$$F_{\mathtt{prop}}(\mathbf{x})\,\|\nabla T(\mathbf{x})\| = 1, \tag{2.10}$$

where the subscript of $F_{\mathtt{prop}}$ indicates that the speed is a normal propagation speed independent of direction, and the Hamiltonian is

$$H(\mathbf{x},\mathbf{p}) = F_{\mathtt{prop}}(\mathbf{x})\|\mathbf{p}\|.$$

Inserting the Hamiltonian into the ODEs (2.6) and (2.7) representing the characteristics, we obtain

$$\begin{aligned}\frac{d\mathbf{x}}{dt} &= F_{\mathtt{prop}}(\mathbf{x})\,\frac{\mathbf{p}}{\|\mathbf{p}\|},\\ \frac{d\mathbf{p}}{dt} &= -\nabla F_{\mathtt{prop}}(\mathbf{x})\,\|\mathbf{p}\|.\end{aligned}$$

Recall from Sect. 2.2.2 that the characteristic lines in the isotropic case follow the gradient lines in the normal direction of the front, and that gradient lines and characteristics coincide.

The first ODE above defines the group velocity vector and has the same direction as the outward normal direction. Suppose that sediments are deposited on horizontal layers. Then, since the direction of the velocity vector is orthogonal to the geological layers, we can relate the speed $F_{\mathtt{prop}}$ to the sedimentation rate at the time of the geological layer's deposition and relate traveltime T to geological time. However, to make a direct correspondence to sedimentation rate and geological time, compaction and other geomechanical and physical processes must also be accounted for. From the second ODE above, we see that if $F_{\mathtt{prop}}$ is constant, then $\mathbf{p}$ is also constant along characteristics in both direction and magnitude. If $F_{\mathtt{prop}}$ varies, for example to reflect varying sedimentation rate, then $F_{\mathtt{prop}}(\mathbf{x})$ must be constant along any level set Γ_t for the

propagating front to resemble parallel layers of a Class 1B fold. The propagation speed can then be recasted to a one-dimensional function $F_{\texttt{prop}}(s)$ along a characteristic, and the characteristic curve reparameterized accordingly to $\mathbf{x}(s)$. We assume that s is a spatial variable that corresponds to arc length such that $\|d\mathbf{x}(s)/ds\| = 1$. Since s is constant along a level set Γ_t and the characteristic points in the gradient direction $\mathbf{p} = \nabla T$, this also implies that $\nabla s = \mathbf{p}/\|\mathbf{p}\|$, and since s is a fixed parameterization, $F_{\texttt{prop}}(s)$ is the same along all characteristics. As $F_{\texttt{prop}}(s)$ varies by s, the gradient $\nabla F_{\texttt{prop}}(\mathbf{x})$ is always directed along the same characteristic since

$$\nabla F_{\texttt{prop}}(\mathbf{x}(s)) = \frac{dF_{\texttt{prop}}(s)}{ds}\,\nabla s = \frac{dF_{\texttt{prop}}(s)}{ds}\frac{\mathbf{p}}{\|\mathbf{p}\|}.$$

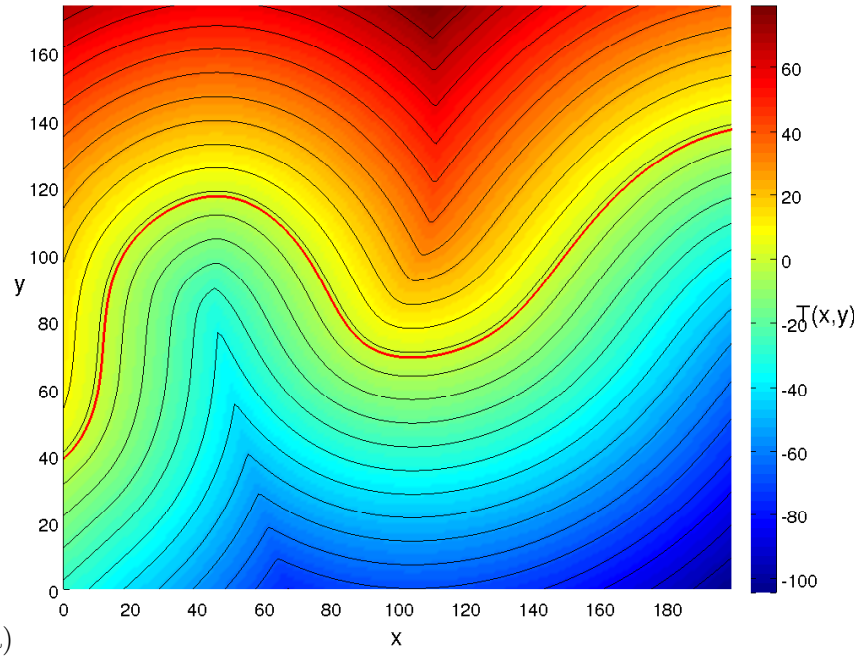
Then, by the second ODE, the gradient direction is constant along the characteristic, and the characteristics are also straight lines under these assumptions.

In 2D the dip angle is

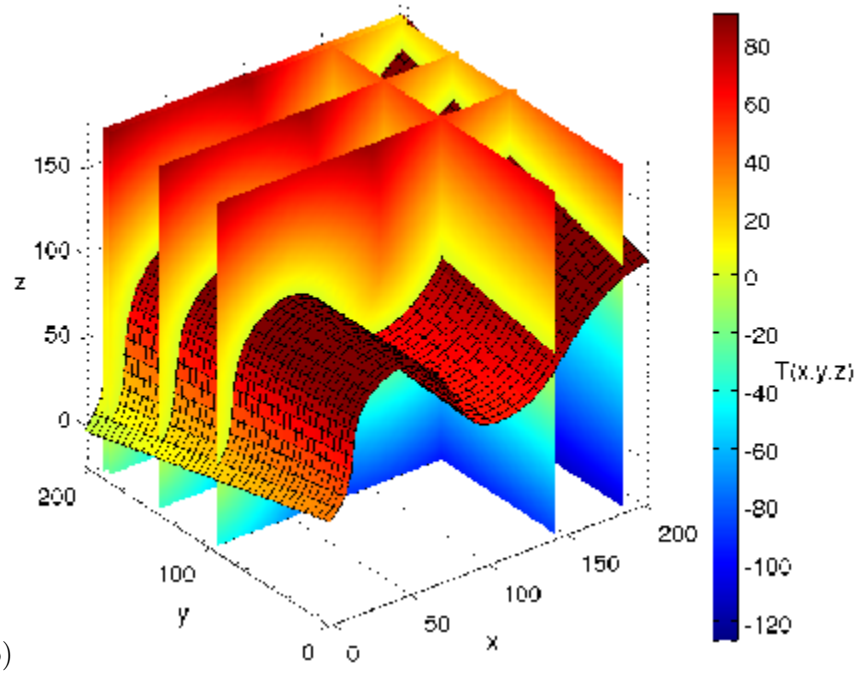
$$\alpha_{\texttt{dip}} = \texttt{sign}(T_x)\cos^{-1}\left(\frac{T_y}{\|\nabla T\|}\right),$$

where $\nabla T = (T_x, T_y)$. Since the gradient direction is constant along a characteristic, the dip angle is also constant along the characteristic. Thus, dip isogons and characteristics coincide. For parallel folds the dip isogons between two layers have equal lengths, since this corresponds to the orthogonal layer thickness. If the corresponding level sets of T of the two layers are Γ_{t_1} and Γ_{t_2} and $F_{\texttt{prop}}$ is constant, the layer thickness is $F_{\texttt{prop}}(t_2 - t_1)$.

Given the boundary condition $T = 0$ on Γ_0, the fast marching method (Sethian, 1996) can be applied directly to solve Eq. (2.10) numerically on a finite difference grid. Fig. 2.4 shows examples from fast marching in 2D and 3D with $F_{\texttt{prop}}(\mathbf{x}) = 1$ such that T represents the distance field relative to a parametric curve in 2D and to a parametric surface in 3D. We notice the solution in the core of the fold where parallelism breaks down and where the solution is clearly not smooth. This is also where characteristic curves of the PDE (and thus dip isogons) meet from different directions. Although ∇T vanishes in these regions, the fast marching method ensures that a unique solution of Eq. (2.10) is found on the finite difference grid by the principles of viscosity solution of Hamilton-Jacobi equations as introduced in Crandall and Lions (1983). Note that T has different signs on each side of Γ_0. In the 2D case we define the positive side to the left of Γ_0, as seen when walking along the curve in the parameter direction. A similar convention is used in the 3D case relative to a parametric surface, where the positive side is in the normal direction of the surface.



(a)



(b)

Fig. 2.4: Parallel Class 1B folds in 2D and 3D represented as time of arrival fields T. Level sets of T are shown in the 2D plot in (a).

2.3.2 Class 2 Fold (Similar)

Let $\mathbf{a}$ be a unit vector representing the axial direction of a similar fold. In the front propagation context we want to translate the level sets Γ_t in the $\mathbf{a}$ direction such that Γ_{t_1} and Γ_{t_2} at two time steps are "similar" in shape, with no thickness variation between Γ_{t_1} and Γ_{t_2} in the direction of $\mathbf{a}$. This simple motion could, of course, be represented for each time step as a graph in a Cartesian system with the abscissa orthogonal to $\mathbf{a}$, but we want to keep this in the context of front propagation to develop a unified mathematical model for all fold classes. The motion can be governed by an underlying advection field in the direction of $\mathbf{a}$ and with constant magnitude on the front Γ_t. We interpret this as a velocity field

$$\psi_{\mathtt{adv}}(\mathbf{x})\,\mathbf{a},$$

where $\psi_{\mathtt{adv}}$ is the advection speed in the direction of $\mathbf{a}$. The advection speed component $F_{\mathtt{adv}}$ normal to the front depends on the direction of the front and can be expressed by the scalar product

$$F_{\mathtt{adv}}\left(\mathbf{x},\mathbf{n}\right) = \psi_{\mathtt{adv}}(\mathbf{x})\left(\mathbf{a}\cdot\mathbf{n}\right).$$

Since the speed in the outward normal direction of the propagating front depends on direction, this is an anisotropic front propagation, as opposed to the isotropic case that models the Class 1B fold. Inserting the speed in the normal direction into Eq. (2.2) and rearranging, we obtain

$$\psi_{\mathtt{adv}}(\mathbf{x})\left(\mathbf{a}\cdot\nabla T(\mathbf{x})\right) = 1. \tag{2.11}$$

This is a linear PDE, as opposed to the non-linear PDE that models Class 1B folds. If the speed $\psi_{\mathtt{adv}}(\mathbf{x}) = 1$, then T represents a "distance field", where distances are measured in the direction of $\mathbf{a}$.

Since the advection speed $F_{\mathtt{adv}}$ normal to the front must always be positive for the motion to be monotonic in the outward direction,

$$\cos^{-1}\left(\mathbf{a}\cdot\mathbf{n}\right) < \pi/2$$

everywhere along the front. This must also be satisfied for Γ_0 representing the lower boundary. Of course, this must also be satisfied for the front propagation to have geological relevance to similar folding, but we will see below that when advection is combined with normal propagation to model other fold classes, $F_{\mathtt{adv}}$ can be less than zero, as long as the total speed in the normal direction is positive. Fig. 2.5 illustrates the Class 2 fold with $\psi_{\mathtt{adv}} = 1.0$ and axial direction $\mathbf{a} = (-0.2, 1.0)$.

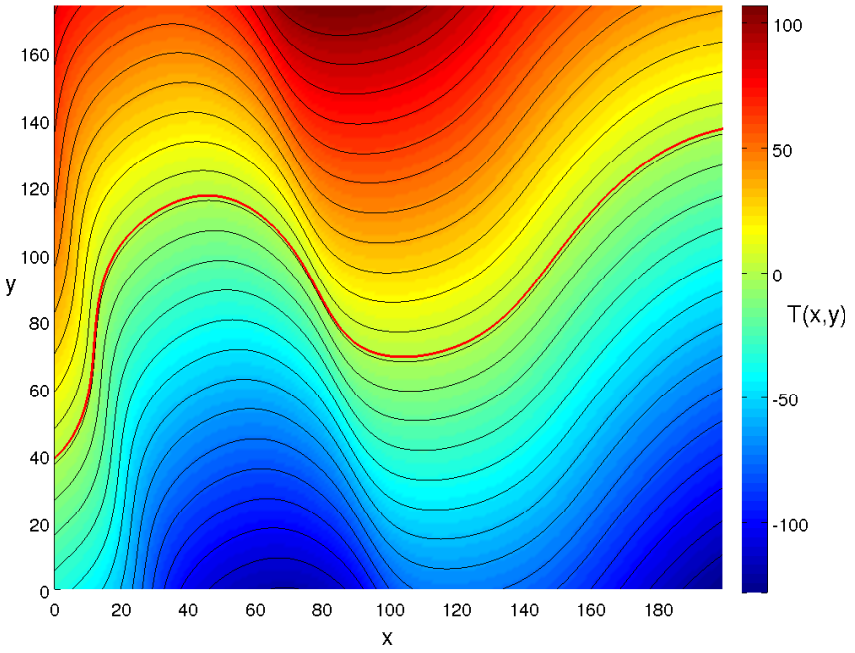


Fig. 2.5: Similar Class 2 fold represented as a time of arrival field with $\psi_{\mathtt{adv}} = 1.0$ and $\mathbf{a} = (-0.2, 1.0)$.

For similar folds, the dip isogons along which the dip angle and the direction $\mathbf{n}$ are constant are parallel and in the direction of $\mathbf{a}$. This is also the direction of the characteristics of the linear PDE (2.11), since the solution of $T(\mathbf{x})$ along curves parallel to $\mathbf{a}$ depends only on other values along that same curve,

$$T(\mathbf{x}(s)) = \int \frac{1}{\psi_{\mathtt{adv}}(s)} ds.$$

Here we use the same spatial parameterization s along the characteristic curve as that introduced for the parallel fold in Sect. 2.3.1.

The system of ODEs representing the characteristics becomes

$$\frac{d\mathbf{x}}{dt} = \psi_{\mathtt{adv}}(\mathbf{x})\,\mathbf{a}, \tag{2.12}$$

$$\frac{d\mathbf{p}}{dt} = -\nabla\psi_{\mathtt{adv}}(\mathbf{x})\,(\mathbf{a}\cdot\mathbf{p}), \tag{2.13}$$

$$\tag{2.14}$$

and the Hamiltonian is

$$H(\mathbf{x},\mathbf{p}) = \psi_{\mathtt{adv}}(\mathbf{x})\,(\mathbf{a}\cdot\mathbf{p}).$$

As for the parallel Class 1B fold, the dip isogons between two layers that correspond to level sets Γ_{t_1} and Γ_{t_2} have equal lengths $\psi_{\mathtt{adv}}(t_2 - t_1)$ if the advection speed $\psi_{\mathtt{adv}}$ is constant. This corresponds to layer thickness in the axial direction. In addition, when $\psi_{\mathtt{adv}}$ is constant, the advection speed $F_{\mathtt{adv}}$ normal to the front depends only on the direction of the front. In terms of geometric optics (Sect. 2.2.2), $\psi_{\mathtt{adv}}$ is the group speed, $\psi_{\mathtt{adv}}\,\mathbf{a}$ is the group velocity vector, and $F_{\mathtt{adv}}$ is the phase speed. Since the phase speed depends only on direction, a similar fold under these restrictions relates to a homogeneous anisotropic medium in geometric optics.

2.3.3 Class 1C Fold

The remaining three fold classes in Fig. 2.1, Class 1A, Class 1C, and Class 3, are best analyzed by starting with the Class 1C fold. From the discussion above, Class 1C is "between" Class 1B and Class 2, with regard to both thickness variations of the limbs and convergence of the dip isogons. More specifically, while Class 2 has parallel dip isogons, Class 1C has convergent dip isogons (as seen when traced from the outer toward the inner arc of a fold), but they are less convergent than dip isogons of Class 1B. In addition, while Class 1B has constant layer thickness in the normal direction, Class 1C has thinning of the limbs relative to that of the hinge zone, but the thinning is less than for Class 2. As such, Class 2 and Class 1B represent two extremes of Class 1C, with regard to both thickness variations of the limbs and convergence of the dip isogons. So, to model this fold as a propagating front, we construct a speed function with contributions from both normal propagation and advection. In the outward normal direction, the combined speed is

$$\begin{aligned} F(\mathbf{x}, \mathbf{n}) &= F_{\mathtt{prop}}(\mathbf{x}) + F_{\mathtt{adv}}(\mathbf{x}, \mathbf{n}) \\ &= F_{\mathtt{prop}}(\mathbf{x}) + \psi_{\mathtt{adv}}(\mathbf{x})\ (\mathbf{a} \cdot \mathbf{n}), \end{aligned} \tag{2.15}$$

with positive $F_{\mathtt{prop}}$ and $\psi_{\mathtt{adv}}$. We note that this is an anisotropic motion with a speed function that depends on the direction $\mathbf{n} = \frac{\nabla T}{\|\nabla T\|}$ of the front. Inserting for F into the Hamilton-Jacobi equation Eq. (2.2) and rearranging, we obtain the non-linear PDE

$$F_{\mathtt{prop}}(\mathbf{x})\|\nabla T(\mathbf{x})\| + \psi_{\mathtt{adv}}(\mathbf{x})\,(\mathbf{a} \cdot \nabla T(\mathbf{x})) = 1. \tag{2.16}$$

We observe that the two terms on the left-hand side come from Eq. (2.10) and (2.11), representing, respectively, Class 1B and Class 2. Fig. 2.6 illustrates the Class 1C fold with $F_{\mathtt{prop}} = 1.0$, $\psi_{\mathtt{adv}} = 1.0$, and axial direction $(-0.2, 1.0)$. The system of ODEs representing the characteristics of Eq. (2.16) are then also

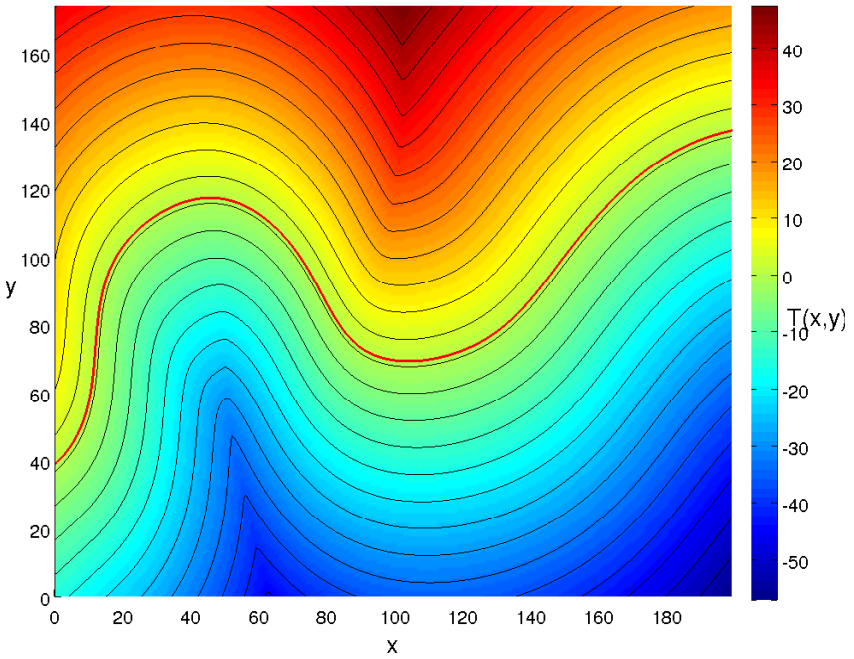


Fig. 2.6: Class 1C fold represented as a time of arrival field with $F_{\mathtt{prop}} = 1.0$, $\psi_{\mathtt{adv}} = 1.0$, and axial direction $\mathbf{a} = (-0.2, 1.0)$.

sums of terms from the characteristics of Class 1B and Class 2,

$$\frac{d\mathbf{x}}{dt} = F_{\mathtt{prop}}(\mathbf{x})\frac{\mathbf{p}}{\|\mathbf{p}\|} + \psi_{\mathtt{adv}}(\mathbf{x})\,\mathbf{a}, \tag{2.17}$$

$$\frac{d\mathbf{p}}{dt} = -\nabla F_{\mathtt{prop}}(\mathbf{x})\,\|\mathbf{p}\| - \nabla\psi_{\mathtt{adv}}(\mathbf{x})\,(\mathbf{a}\cdot\mathbf{p}), \tag{2.18}$$

$$\tag{2.19}$$

and the Hamiltonian becomes

$$H(\mathbf{x},\mathbf{p}) = F_{\mathtt{prop}}(\mathbf{x})\|\mathbf{p}\| + \psi_{\mathtt{adv}}(\mathbf{x})(\mathbf{a}\cdot\mathbf{p}).$$

Similarly as for Class 1B and Class 2, the normal propagation speed and advection speed can be recasted to one-dimensional functions $F_{\mathtt{prop}}(s)$ and $\psi_{\mathtt{adv}}(s)$, where s is a spatial parameter representing arc length. The characteristic curves can be reparameterized accordingly. Then $F_{\mathtt{prop}}(s)$ and $\psi_{\mathtt{adv}}(s)$ can vary along the gradient lines and in the direction of $\mathbf{a}$, respectively. But for the gradient lines and the characteristics to be straight lines, it follows from Eq. (2.17) and (2.18) that one must require

$$\frac{F_{\mathtt{prop}}(s)}{\psi_{\mathtt{adv}}(s)} = \text{constant}. \tag{2.20}$$

Then the dip angle is constant along the characteristic, and the dip isogons and characteristics coincide.

The vector sum in Eq. (2.17), which defines the group velocity and the tangent vector along the characteristic, can be given a precise geometric interpretation in agreement with the discussion in the introduction to this section. The first vector in the sum has the same direction as the dip isogons of the Class 1B fold, and the second vector has the same direction as the dip isogons of the Class 3 fold (Fig. 2.7). Class 1B has convergent dip isogons and Class 3 has parallel dip isogons. Thus, since $F_{\mathtt{prop}}$ and $\psi_{\mathtt{adv}}$ are both positive, the characteristics expressed by the vectors in Eq. (2.17) are less convergent than the characteristics of the Class 1B fold, and as long as $F_{\mathtt{prop}}$ is greater than zero, the characteristics are indeed convergent. As $F_{\mathtt{prop}}$ becomes smaller relative to $\psi_{\mathtt{adv}}$, the characteristics become less convergent, and in the limit as $F_{\mathtt{prop}} \to 0$, the characteristics become parallel and we obtain Eq. (2.11) for the similar Class 2 fold. As $\psi_{\mathtt{adv}}$ becomes smaller relative to $F_{\mathtt{prop}}$, we get the isotropic case in the limit with the eikonal equation (2.10) for the parallel Class 1B fold.

2.3.4 Class 1A Fold

From the discussion above on the vector sum in Eq. (2.17) representing the tangent vector of the characteristics, we can now conclude that if $F_{\mathtt{prop}}$ is greater than zero and $\psi_{\mathtt{adv}}$ is less than zero, the characteristics, and thus the dip isogons, are more convergent than those of the Class 1B fold. Fig. 2.8 illustrates this situation. The total speed in the normal direction must be positive for the front propagation to be monotonic, so from Eq. (2.15) the restriction on the advection speed along the front is

$$\psi_{\mathtt{adv}} > -\frac{F_{\mathtt{prop}}}{\mathbf{a}\cdot\mathbf{n}}. \tag{2.21}$$

From here we follow the same line of arguments as above for Class 1C. In particular, we find that the characteristics are straight lines along which the gradient direction is constant under the restriction of Eq. (2.20). Thus, the characteristics and dip isogons coincide. Moreover, if Eq. (2.21) is satisfied everywhere on the initial front Γ_0, it is satisfied along all characteristics and therefore in the whole domain. We conclude that the propagating front governed by the static Hamilton-Jacobi Equation (2.16), with $F_{\mathtt{prop}}$ greater that zero, $\psi_{\mathtt{adv}}$ less than zero, and with the above restriction on $\psi_{\mathtt{adv}}$, represents the Class 1A fold (Fig. 2.9).

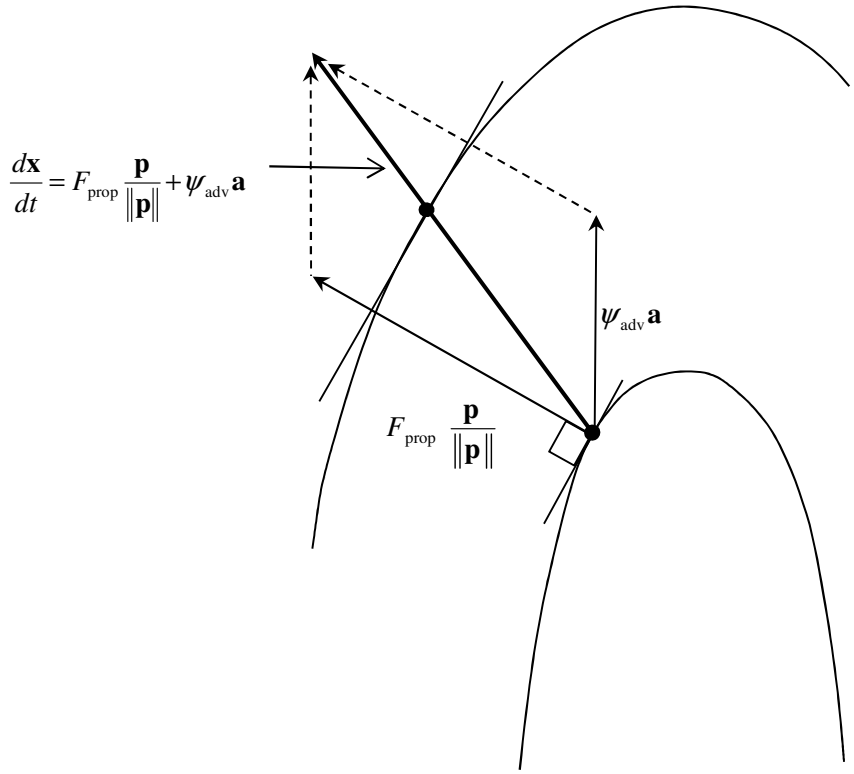


Fig. 2.7: Illustration of the vector sum in Eq. (2.17) for the Class 1C fold that sums to the group velocity in the direction of the dip isogon. Both $F_{\texttt{prop}}$ and $\psi_{\texttt{adv}}$ are positive.

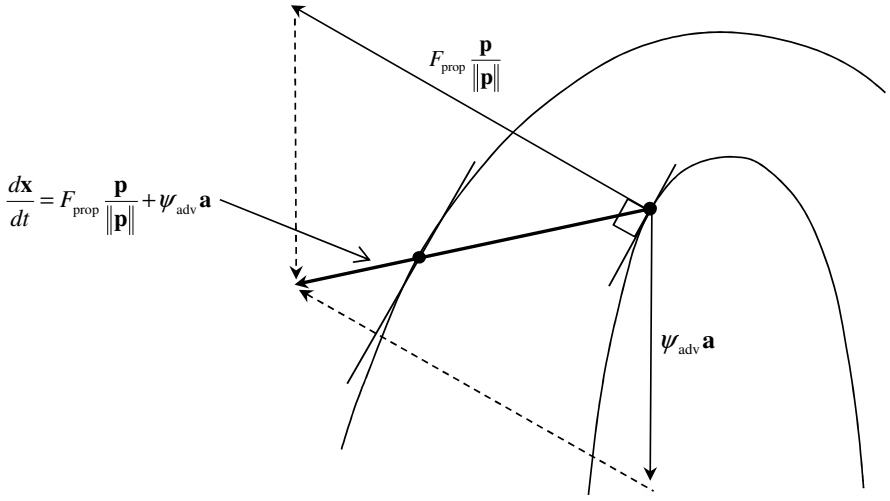


Fig. 2.8: Illustration of the vector sum in Eq. (2.17) for the Class 1A fold that sums to the group velocity in the direction of the dip isogon. Here $F_{\texttt{prop}}$ is positive and $\psi_{\texttt{adv}}$ is negative.

2.3.5 Class 3 Fold

Similarly as above for Class 1A and Class 1C, we examine the vector sum in Eq. (2.17) representing the tangents of the characteristics. If $F_{\texttt{prop}}$ is less than zero and $\psi_{\texttt{adv}}$ is greater than zero, the characteristics are divergent. Still, the total speed in the normal direction must be positive for the front propagation to be monotonic, so from Eq. (2.15) the restriction on the normal propagation speed is

$$F_{\texttt{prop}} > -(\psi_{\texttt{adv}}\, \mathbf{a}) \cdot \mathbf{n}. \tag{2.22}$$

The rest of the discussion above also applies directly here. We conclude that Eq. (2.16) with $F_{\texttt{prop}}$ less than zero, $\psi_{\texttt{adv}}$ greater than zero, and with the above restriction on $F_{\texttt{prop}}$ represents the Class 3 fold. Fig. 2.10 shows an example with the same initial front Γ_0 as was used for the other fold classes.

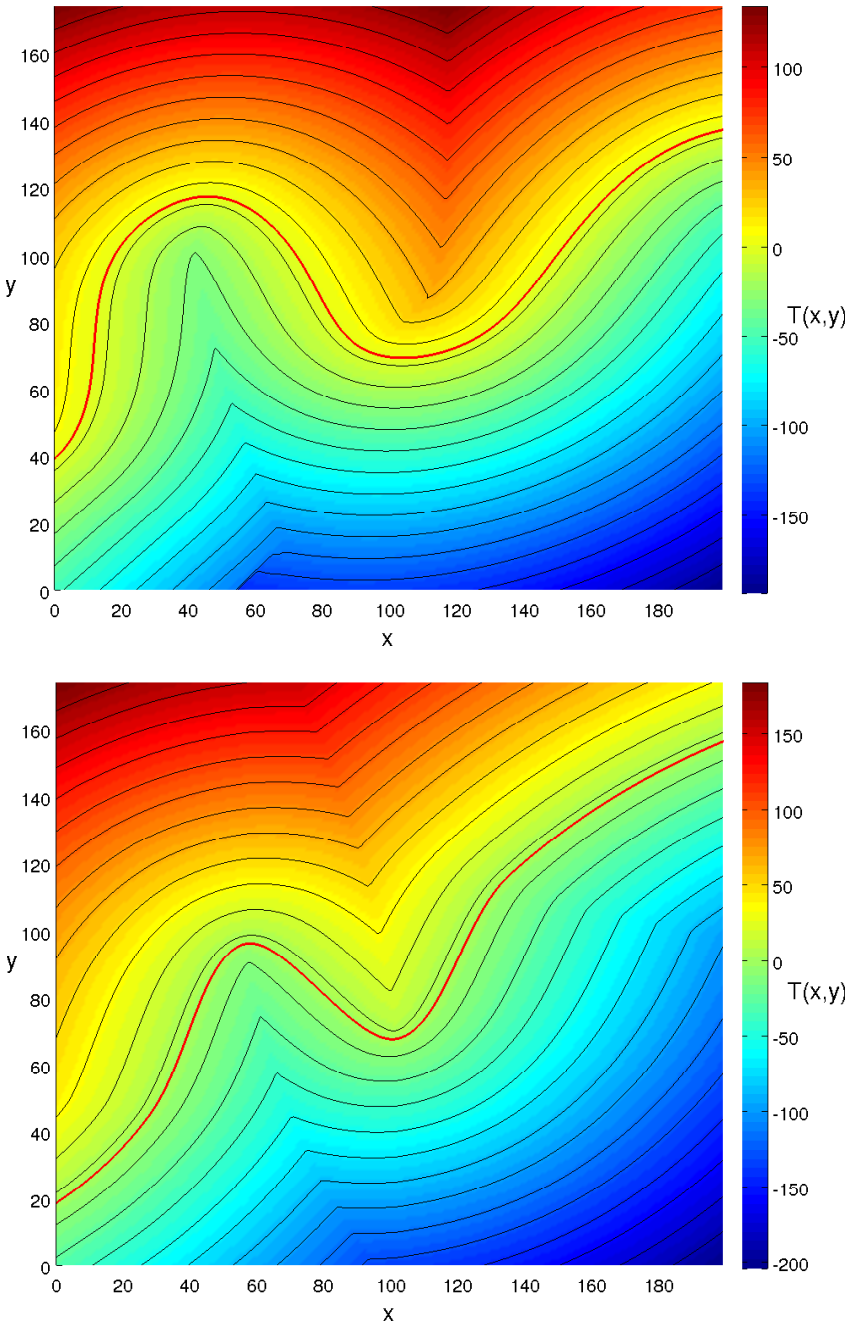


Fig. 2.9: Class 1A folds represented as time of arrival fields with $F_{\texttt{prop}} = 1.0$, $\psi_{\texttt{adv}} = -0.5$, and axial direction $\mathbf{a} = (-0.2, 1.0)$.

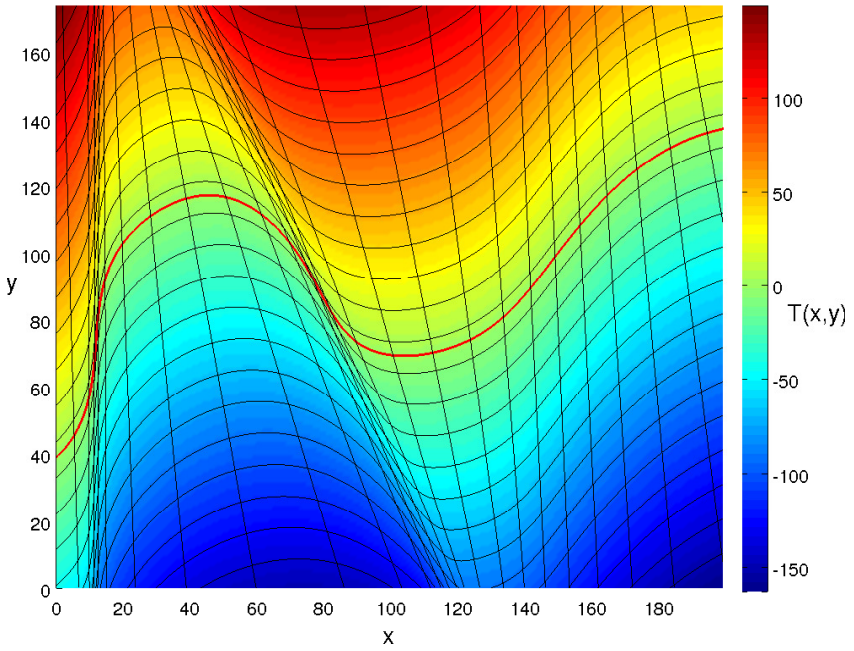


Fig. 2.10: Class 3 fold represented as a time of arrival field with $F_{\texttt{prop}} = -0.2$, $\psi_{\texttt{adv}} = 1.0$, and axial direction $\mathbf{a} = (-0.2, 1.0)$. A set of (divergent) dip isogons is also shown.

2.4 The Unified Mathematical Model for all Fold Classes

We now have a unified mathematical model for all five fold classes in Fig. 2.1 represented by the Hamiltonian

$$H(\mathbf{x}, \nabla T) = F_{\texttt{prop}}(\mathbf{x})\|\nabla T\| + \psi_{\texttt{adv}}(\mathbf{x})\,(\mathbf{a}\cdot\nabla T)\,.$$

The boundary value problem for the front propagation modeling the fold classes is

$$\begin{aligned} &F_{\texttt{prop}}(\mathbf{x})\|\nabla T(\mathbf{x})\| + \psi_{\texttt{adv}}(\mathbf{x})\,(\mathbf{a}\cdot\nabla T(\mathbf{x})) = 1, \\ &\text{given}\ \ T = t_0 \text{ on } \Gamma_0. \end{aligned} \tag{2.23}$$

In the previous examples we have used the boundary condition $t_0 = 0$ on Γ_0, but t_0 can be any constant. For the sake of completeness, we also repeat the

Fold class	$F_{\texttt{prop}}$	$\psi_{\texttt{adv}}$	Curvature
Class 1A	> 0	< 0	$\kappa_1 > \kappa_2$
Class 1B, Parallel	> 0	0	$\kappa_1 > \kappa_2$
Class 1C	> 0	> 0	$\kappa_1 > \kappa_2$
Class 2, Similar	0	> 0	$\kappa_1 = \kappa_2$
Class 3	< 0	> 0	$\kappa_1 < \kappa_2$

Table 2.2: Characterization of the fold classes by the sign of the normal propagation speed $F_{\texttt{prop}}$ and the advection speed $\psi_{\texttt{adv}}$. The curvatures κ_1 and κ_2 refer to the inner and the outer arcs, respectively.

ODEs representing the characteristic curves (and the dip isogons),

$$\frac{d\mathbf{x}}{dt} = \nabla_{\mathbf{p}} H = F_{\texttt{prop}}(\mathbf{x}) \frac{\mathbf{p}}{\|\mathbf{p}\|} + \psi_{\texttt{adv}}(\mathbf{x})\, \mathbf{a}, \tag{2.24}$$

$$\frac{d\mathbf{p}}{dt} = -\nabla_{\mathbf{x}} H = -\nabla F_{\texttt{prop}}(\mathbf{x})\, \|\mathbf{p}\| - \nabla \psi_{\texttt{adv}}(\mathbf{x})\, (\mathbf{a} \cdot \mathbf{p}). \tag{2.25}$$

$$\tag{2.26}$$

Here we have also included the general form of the characteristics for the general static anisotropic Hamilton-Jacobi equation (Sect. 2.2.2).

Different fold classes are modeled by varying the sign of the normal propagation speed $F_{\texttt{prop}}$ and the advection speed $\psi_{\texttt{adv}}$, as summarized in Table 2.2. Comparing this table with Table 2.1 in Sect. 2.1.2 reveals the relation between the speed components $F_{\texttt{prop}}$ and $\psi_{\texttt{adv}}$ in the Eulerian model and thickness variation t'_α and isogon angle ϕ in Ramsay's classification model. Moreover, the five folding regimes can be continuously spanned by varying the magnitude of the two speed components. The total speed in the normal direction must always be positive, so when $\psi_{\texttt{adv}}$ or $F_{\texttt{prop}}$ is less than zero, they are restricted by Eq. (2.21) and (2.22), respectively.

The PDE in Eq. (2.23) can also be interpreted as a generalized eikonal equation used in acoustics for traveltime modeling (Kornhauser, 1953). For example, we may think of Γ_0 as a point source that is moving (or advected) in a medium in the direction of $\mathbf{a}$ with speed $\psi_{\texttt{adv}}$, and radiating sound waves with speed $F_{\texttt{prop}}$ in all directions. Then the solution $T(\mathbf{x})$ represents the traveltime of the sound from Γ_0 to $\mathbf{x}$. In this simple case, when Γ_0 is just a point source and $F_{\texttt{prop}}$ and $\psi_{\texttt{adv}}$ are constant, the solution for T can be found analytically.

In Chapter 3 we discretize the boundary value problem of Eq. (2.23) by upwind finite differences and compute T on a regular grid by a variant of the fast marching method inspired by Sethian and Vladimirsky (2003). Variants

of the fast sweeping method (Kao *et al.*, 2005; Qian *et al.*, 2007; Zhao, 2005), or the fast iterative method (Jeong and Whitaker, 2008) could also be used to compute T.

Fig. 2.11 shows the result from modeling folds and simulating property distribution over the folds on the surface of a road cut. Seven interfaces have been digitized in the photo in (a). Then $F_{\texttt{prop}}$ and $\psi_{\texttt{adv}}$ values are set to model the folding regimes between the interfaces. The folds are of type Class 1C and Class 2 (similar). The Class 1C folds have small $F_{\texttt{prop}}/\psi_{\texttt{adv}}$ ratios that make them almost Class 2 type. Properties (here representing image intensities) are distributed from each interface in the directions of the characteristics of the underlying PDE. A weighted average of properties distributed from the interface below and above each point in the computational grid is then set as the property value. This scheme for property distribution extends easily to 3D, where the interfaces are surfaces (Petersen *et al.*, 2007). We return to this in Chapter 4. The modeling was done in Statoil's Compound Earth Simulator (CES).

Metric properties such as distances, gradients, curvature, and their spatial variations can also be consistently computed by the numerical scheme. In particular, the gradient $\nabla T(\mathbf{x})$ uniquely represents the dip and strike of a fold at any point $\mathbf{x}$ in 3D space. Fig. 2.12 shows a 2D example with the gradient field components T_x and T_y corresponding to the Class 1C fold in Fig. 2.6.

Curvature also plays an important role in the description of folded layers. In the two-dimensional case, the scalar curvature is the divergence of the unit normal vector to the front,

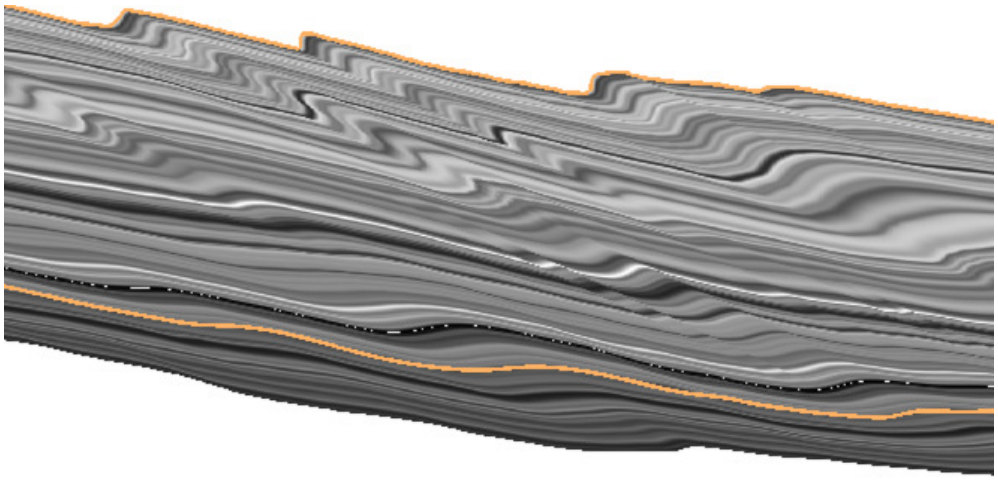
$$\kappa = \nabla \cdot \mathbf{n} = \nabla \cdot \frac{\nabla T}{\|\nabla T\|} = \frac{T_{xx}T_y^2 - 2T_xT_yT_{xy} + T_{yy}T_x^2}{(T_x^2 + T_y^2)^{3/2}},$$

where we have use the notation $T_{\mu\nu} = \partial^2 T/(\partial\mu\partial\nu)$. The right hand side is easily obtained by applying the chain rule (see details in Appendix C). In 3D the Gaussian curvature and the mean curvature can be calculated similarly from the first and second derivatives (Goldman, 2005). The second derivatives can be computed in the same upwind fashion as the first derivatives, though a numerical scheme of at least second-order is required to express the second derivatives consistently.

2.4.1 Characteristics and Dip Isogons Revisited

Recall that the value of T at a point $\mathbf{x}$ depends only on the values of T along the characteristic passing through $\mathbf{x}$. We have imposed restrictions on the speed

(a)



(b)

Fig. 2.11: (a) Photo of a vertical planar road cut with metamorphosed mica schists of Silurian age, and (b) the result of fold modeling and simulating property distribution over the folds in (a).

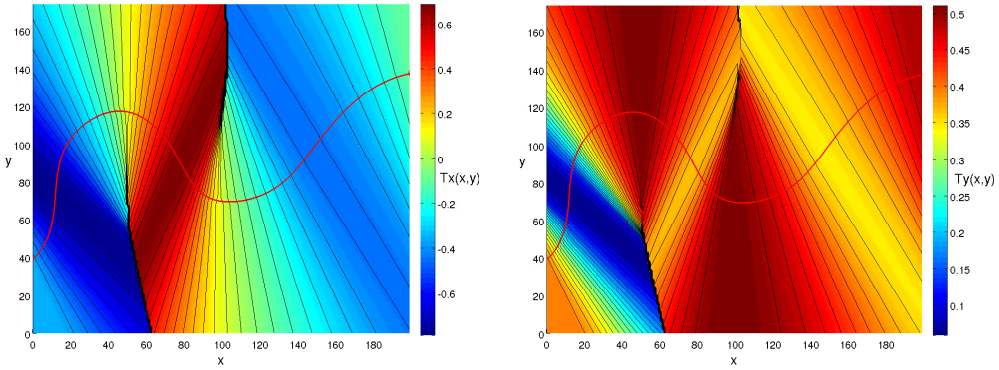


Fig. 2.12: Gradient field $\nabla T = (T_x, T_y)$ of the Class 1C fold in Fig. 2.6 (T_x left and T_y right).

functions to obtain an exact match between the characteristics and dip isogons of the fold classes defined in the classical literature in structural geology. Thus, the characteristics are the key to modeling different folding regimes. We focus more attention on characteristics here for several reasons.

(i) The exact match between characteristics and dip isogons links the Hamilton-Jacobi mathematical model directly to an intuitive geometrical model (Fig. 2.7 and 2.8), which makes it easy to derive and analyze metric properties everywhere in the geological volume.

(ii) Characteristics play an important role in the initial critical step of numerical methods, such as the fast marching method (Sethian and Vladimirsky, 2003), the fast sweeping method (Kao *et al.*, 2005; Qian *et al.*, 2007; Zhao, 2005), and the fast iterative method (Jeong and Whitaker, 2008). All these methods start by computing solution values on a finite difference grid in a narrow band around the initial front Γ_0. The solution value for a grid point near Γ_0 can be found by solving the ODE (2.24) for a characteristic that passes through the grid point.

(iii) Characteristics are the key to understanding how the solution of the PDEs develops and propagates downwind from the solution around the initial front Γ_0. As a consequence, characteristics are the key to designing consistent local solvers for the numerical methods mentioned above (Qian *et al.*, 2007; Vladimirsky, 2001).

We return to these properties of the characteristics in the next chapter. Characteristic curves of a hyperbolic PDE can cross each other. Then multivalued solutions can exist, since the solutions of the ODEs along the characteristics can

be different at the crossings. This causes so-called shock waves. In the context of front propagation modeled by the static Hamilton-Jacobi equation (2.23), the characteristics can also meet when they are convergent. Since $T(\mathbf{x})$ is the time of arrival value of the propagating front at $\mathbf{x}$, however, and more specifically the "first arrival time value," T is never multivalued and therefore continuous. On the other hand, the gradient ∇T is not continuous where characteristics meet, for example, in the core of the Class 1B fold in Fig. 2.4(a). In a similar fold characteristics are parallel and never meet, such that both T and ∇T are continuous everywhere.

The system of ODEs by Eq. (2.24) and (2.25) shows how the velocity and gradient develop over time along the characteristics. To see how the solution develops in space along the characteristics, we also give the derivatives with respect to the space variable s, which corresponds to arc length along the characteristic. We first note that the group speed is $v_g = \|\nabla_{\mathbf{p}} H\| = ds/dt$. Applying the chain rule to the ODEs, we get

$$\frac{d\mathbf{x}}{ds} = \frac{\nabla_{\mathbf{p}} H}{\|\nabla_{\mathbf{p}} H\|},$$
$$\frac{d\mathbf{p}}{ds} = -\frac{\nabla_{\mathbf{x}} H}{\|\nabla_{\mathbf{p}} H\|}.$$

In addition, we have

$$\frac{dT}{ds} = \mathbf{p} \cdot \frac{d\mathbf{x}}{ds} = \mathbf{p} \cdot \frac{\nabla_{\mathbf{p}} H}{\|\nabla_{\mathbf{p}} H\|} = \frac{1}{\|\nabla_{\mathbf{p}} H\|},$$

where we have used $\mathbf{p} \cdot \nabla_{\mathbf{p}} H = 1$ from Eq. (2.9).

Recall from Sect. 2.1.2 the method by Hudleston (1973) for classifying folds based on the isogon angle. The isogon angle ϕ was defined as the angle between the normal vector and the dip isogon. If we disregard the sign of ϕ, the isogon angle can be expressed as the angle between the gradient vector and the group velocity vector,

$$\cos\phi = \frac{\nabla T \cdot \nabla_{\mathbf{p}} H}{\|\nabla T\| \|\nabla_{\mathbf{p}} H\|} = \frac{1}{\|\nabla T\| \|\nabla_{\mathbf{p}} H\|}.$$

In the Eulerian model, this measure can be calculated in the whole 3D space when an approximation for the gradient ∇T is known. We observe that in the isotropic case with $\psi_{\mathtt{adv}} = 0$, we get $\|\nabla T\| \|\nabla_{\mathbf{p}} H\| = 1$ and the isogon angle is zero. Fig. 2.13 shows the isogon angle for a Class 1C fold.

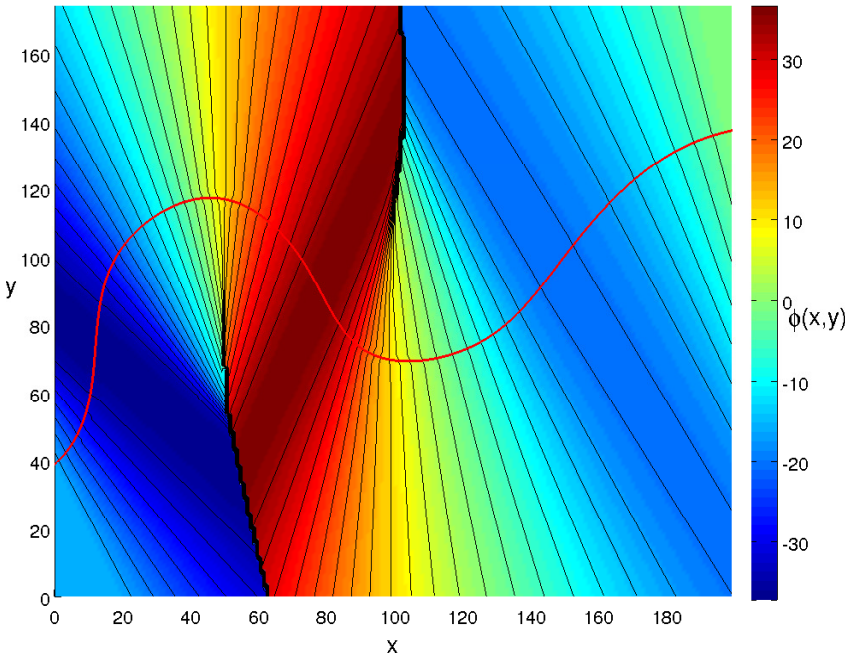


Fig. 2.13: Isogon angle ϕ in degrees for a Class 1C fold with $F_{\texttt{prop}} = 1$, $\psi_{\texttt{adv}} = 1$, and $\mathbf{a} = (-0.2, 1.0)$. The time of arrival field is shown in Fig. 2.6.

2.5 Summary

A mathematical framework for modeling folds in structural geology is proposed. The mathematical model is founded on a front propagation analogy by which a continuum of layer boundaries of a fold is regarded as a propagating front starting from a reference layer boundary. Each of the five fold classes from the classical literature of structural geology is represented in an Eulerian context as a time of arrival field T, which is the solution of a static Hamilton-Jacobi equation. We pay special attention to dip isogons of folds, and we chose to align dip isogons with the characteristic curves of the Hamilton-Jacobi equation when we derive the mathematical model. This links the Hamilton-Jacobi mathematical model directly to an intuitive geometrical model which makes it easy to derive and analyze metric properties in the geological volume. Characteristics also play an important role in the numerical framework we develop in the next chapter for computing the T-field and other metric properties.

In the next chapter we derive a numerical scheme and an algorithmic framework based on a variant of the fast marching method to compute the T-field representing the geological folds. In Chapter 4 we extend the mathemati-

cal and numerical frameworks, developed in this chapter and in Chapter 3, and develop novel methods for data restoration, model reconstruction and property distribution. These methods are applied to support processes in the dR/mR-workflows in CES. Chapter 4 also pay attention to the physical problem of folding, including strain states set up during fold formation.

Chapter 3

A Numerical Framework for Modeling Folds in Structural Geology[1]

In this chapter a rigorous numerical framework is developed for the Hamilton-Jacobi formulation for fold modeling presented in Chapter 2. The static Hamilton-Jacobi equation, which models all the fold classes from the classical literature, is discretized by upwind finite differences and a dynamic stencil construction. This forms the basis of numerical solution by finite difference solvers such as fast marching and fast sweeping methods. A new robust and accurate scheme for initialization of finite difference solvers for the static Hamilton-Jacobi equation is also derived. In the context of the mathematical model derived in Chapter 2, fold classification based on dip isogons is particularly relevant. This specific interest is present since dip isogons coincide with the characteristic curves of the static Hamilton-Jacobi equation. The numerical scheme uses the direction of the characteristics for the upwind stensil construction, and thus propagates the solution of the T-field along the dip isogons of the fold. This framework, together with extensions presented in Chapter 4, serve as the numerical engine in the dR/mR-workflows in CES for process based data restoration and model reconstruction outlined in Chapter 1. A numerical example from CES is presented in the end of this chapter based on seismic data collected from the Karama Block in the North Makassar Strait outside Sulawesi.

---

[1]This chapter is based on Hjelle *et al.* (2013)

In Sect. 3.1 we first outline the fast marching algorithm that we use for solving the Hamilton-Jacobi problem. In particular we derive a robust and accurate scheme for the initialization of fast marching methods and other finite difference solvers in Sect. 3.1.1 and 3.1.2. Then we discretize the Hamilton-Jacobi formulation by upwind finite differences and derive a numerical scheme for the solution on a finite difference grid (Sect. 3.1.3, 3.1.4 and 3.1.5). In Sect. 3.2 we present numerical examples of fold modeling with some discussion.

3.1 Numerical Scheme for Fold Modeling

In this section we discretize the boundary value problem in Eq. (2.23) by upwind finite differences and derive a numerical scheme that forms the basis of numerical solution by several methods. In particular, one may apply variants of the fast marching method (Sethian and Vladimirsky, 2003), the fast sweeping method (Kao *et al.*, 2005; Qian *et al.*, 2007; Zhao, 2005), or fast iterative methods (Jeong *et al.*, 2007; Gillberg, 2011). The isotropic front propagation problem in Eq. (2.4), which models the parallel Class 1B fold, can be solved directly and elegantly on a rectangular finite difference grid by the original fast marching method introduced by Sethian (1996). The fast iterative method, which enables implementation on parallel hardware, is also a good choice. The other special case, modeling the Class 2 fold, can be easily solved when the axial direction $\mathbf{a}$ is constant (Sect. 3.1.4).

To solve the fully non-linear problem in Eq. (2.23), we implement a variant of the fast marching method for anisotropic problems, inspired by Sethian and Vladimirsky (2003) and Lin (2003). The basic structure of our algorithm is the same as the original fast marching method. However, the finite difference operators are different since the upwind and causality conditions are more complicated for anisotropic problems, where the directions of the gradients and the characteristics of the PDE do not coincide as in the isotropic case (cf. Eq. (2.24)). The finite difference grid has still a rectangular pattern, but the diagonal directions are used in the stencil construction both for better directional resolution and to guarantee the causality condition. While the finite difference approach is Eulerian in the sense that fold geometry is represented as continua in the whole three-dimensional volume, we also present a Lagrangian approach in Sect. 3.1.1, which is used in the initialization step of the finite difference solvers. The numerical scheme for the initialization is given in Sect. 3.1.2.

For the sake of completeness and as an explanation of terms used later, the fast marching algorithm is outlined here first. Details of the algorithm are

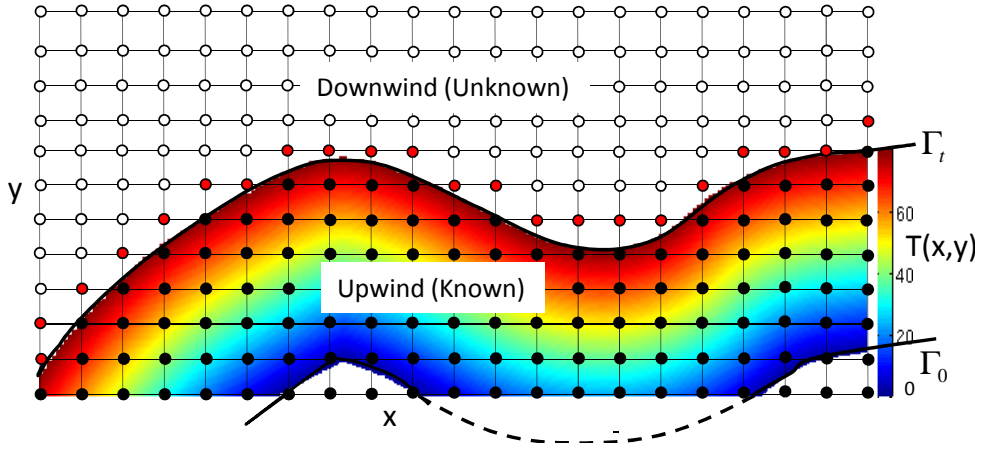


Fig. 3.1: Snapshot of the progress of fast marching at time t, cf. Algorithm 3.1, steps 4–7. Points on the upwind side of Γ_t have got a fixed value and are tagged as *Known*. Points in a narrow band just on the downwind side of Γ_t, filled with red colour, have been computed, tagged as *Trial* and put on the heap. Other points on the downwind side are *Unknown* points, yet to get values.

given in the numerical schemes throughout the chapter. The other algorithms are thoroughly explained in the references given above and are not pursued further here. Although the boundary value problem (2.23) is global, the fast marching algorithm computes all arrival time values in one pass only. This is done by computing grid point values in a narrow band around the propagating front Γ as it evolves in space and time. A snapshot of the progress of the algorithm is illustrated in Fig. 3.1.

Initialization of fast marching starts by tagging all grid points in the domain as *Unknown* (Algorithm 3.1, Step 1). Then, grid points in a small neighbourhood around the initial front Γ_0 are computed (Step 2). This is a critical step that requires high geometric accuracy, since these grid points are used by the finite difference operators in the next step of the algorithm. We use a Lagrangian approach and a fast converging Newton iteration scheme to compute these points, where the characteristic equation (2.24) is solved in each step of the iteration. This is described in detail in Sects. 3.1.1 and 3.1.2. The computed points are tagged as *Known* and will never be re-computed. In the last Step 3 of the initialization, edge connected *Unknown* neighbours of the *Known* points are computed by solving a discretized version of Eq. (2.23). These points are tagged as *Trial* and put on a heap data structure (Sedgewick and Wayne, 2010), where the points are kept sorted on their calculated value. When discretizing Eq. (2.23), we will see in the next sections that only points

Algorithm 3.1 The fast marching algorithm, cf. Fig. 3.1

Initialization:

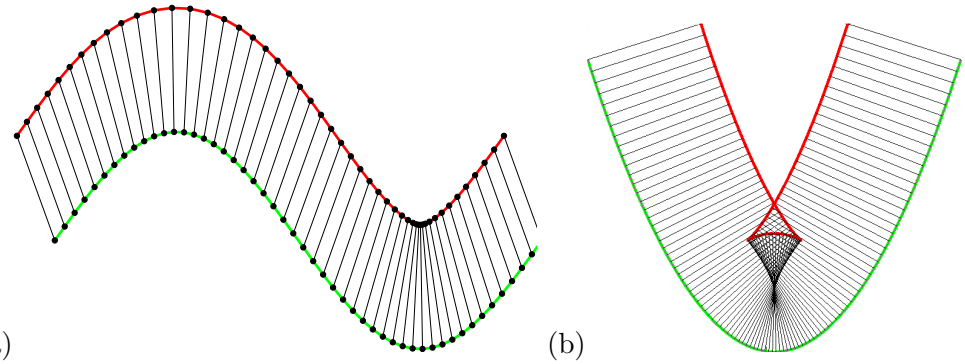
1. Tag all grid points as *Unknown*.
2. Compute grid point values in a small neighbourhood around Γ_0 using Algorithm 3.2 in Sect. 3.1.2. Tag these points as *Known*.
3. Compute all *Unknown* edge connected neighbours of *Known* points by solving the discretized version of Eq. (2.23). Tag these points as *Trial* and put them on the heap.

Loop:

4. Let A be the *Trial* point with the smallest computed value. Remove A from the heap and tag it as *Known*.
5. Tag as *Trial* all edge connected neighbours of A that are *Unknown* and put them on the heap.
6. Re-compute all *Trial* neighbours of A by solving the discretized version of Eq. (2.23).
7. Go to Step 4

on the upwind side are used in the finite difference operators.

The algorithm then proceeds in a loop and computes all points that are not yet *Known* in one pass (Steps 4–7). In each iteration of the loop, the point with the smallest *Trial* value is accepted, tagged as *Known* and removed from the heap. This ordering of accepting computed values is motivated by the fact that T represents first arrival times of the propagating front as it reaches the grid points. Its non-accepted neighbours are computed, or re-computed, and put on the heap if they have not already been stored there. As such, the narrow band with *Trial* values is advanced downwind with the propagating front, with accepted points tagged as *Known* on its upwind side and *Unknown* points on its downwind side, until all points are eventually tagged as *Known* and the heap is empty. The heap operations, removing and inserting a new *Trial* point or reorganizing the heap when a *Trial* point is given a new value, are logarithmic. Therefore, if the discretized domain has N grid points and N_h is the maximum number of trial values on the heap, the fast marching algorithm is of order $O(N \log N_h)$. When applied to fold modeling as proposed here, N_h



(a) (b)

Fig. 3.2: (a): Marker particles marked with • moving upward along characteristics of the PDE from the initial front Γ_0. In this example $F_{\texttt{prop}} = 0.3$, $\psi_{\texttt{adv}} = 0.5$ and axial direction is $(0, 1)$. (b): Swallow-tail solution from the marker particle method with intersecting characteristics.

is small compared to N in most cases, such that the computational complexity is almost linear in N.

3.1.1 Lagrangian Approach

When solving the front propagation problem, there are alternatives to the Eulerian finite difference approach. In this section, we present a Lagrangian method since it will be of practical interest when implementing the boundary condition of Eq. (2.23) in the narrow band of grid points around the initial front Γ_0. The Lagrangian approach can also be used to model individual layers of a fold. The procedure proposed here can be regarded as a variant of so-called marker particle methods (Harlow and Welch, 1965; Sethian, 1985). The principle of marker particle methods is to consider the propagating front as a set of discrete points whose positions at any time represent the propagating front. The technique has been used in a variety of applications, for example, to compute Voronoi diagrams in flow fields (Nishida *et al.*, 2007).

We have already established Eq. (2.24) and (2.25) for the motion of a particle along a characteristic of the Hamilton-Jacobi equation (2.23). Recall that characteristics coincide with dip isogons. In our marker particle model particles are moved along dip isogons governed by these equations. Let Γ_0 be the initial front with a set of marker particles on it. For simplicity, consider the two-dimensional problem where the propagating front is an evolving curve (Fig. 3.2(a)). We want to calculate new positions for the marker particles as they move along the characteristics from Γ_0 at time t_0 to new positions on

the level set Γ_1 at time t_1. The time step $t_1 - t_0$ is small. Assume first that Γ_0 is represented as a smooth parametric curve. Let $\mathbf{x}_i = \mathbf{x}(\tau_i) = (x_i(\tau_i), y_i(\tau_i))$ be the position of a marker particle on Γ_0, where τ_i is the parameter value of $\mathbf{x}_i$ on the curve, and let $\mathbf{n}_i = \mathbf{n}(\tau_i)$ be the outward unit normal vector of Γ_0 at $\mathbf{x}_i$. If Γ_0 has continuous first derivatives, the normal vector can be evaluated from the tangent vector $\dot{\mathbf{x}}(\tau_i) = (x'(\tau_i), y'(\tau_i))$ as

$$\mathbf{n}_i = \frac{(-y'(\tau_i), x'(\tau_i))}{\|\dot{\mathbf{x}}(\tau_i)\|},$$

where $\mathbf{n}_i$ points to the left as seen when walking along Γ_0 in the parameter direction. Alternatively, if Γ_0 is represented by linear segments between marker particles, the normal vector can be computed by central differences from the positions of the marker particles. If the particles are equally spaced, a first order approximation to the unit normal is

$$\mathbf{n}_i = \frac{(-(y_{i+1} - y_{i-1}), (x_{i+1} - x_{i-1}))}{((x_{i+1} - x_{i-1})^2 + (y_{i+1} - y_{i-1})^2)^{1/2}}\,.$$

We utilize the fact that the solution value (arrival time or spatial displacement of a marker particle) of the propagating front in a point depends only on the solution values along the characteristic passing through this point. The position of a marker particle at any time can be found by solving the ODE (2.24), and thus by "moving" the marker particle along a characteristic of the PDE (and along a dip isogon) a time step. If we assume local homogeneity with constant $F_{\texttt{prop}}$, $\psi_{\texttt{adv}}$ and $\mathbf{a}$, the velocity vector in $\mathbf{x}_i$ can be expressed as

$$\mathbf{v}_i = F_{\texttt{prop}}\,\mathbf{n}_i + \psi_{\texttt{adv}}\,\mathbf{a}. \tag{3.1}$$

The new position of the particle at time t_1 becomes

$$\mathbf{x}_g = \mathbf{x}_i + \mathbf{v}_i(t_1 - t_0),$$

and the time of arrival in $\mathbf{x}_g$ is

$$T(\mathbf{x}_g) = t_0 + \frac{\|\mathbf{x}_g - \mathbf{x}_i\|}{\|\mathbf{v}_i\|}. \tag{3.2}$$

The Lagrangian approach is easy to implement using these principles when marker particles are moved along characteristics. Fig. 3.2(a) shows an example of an evolving curve between two time steps. The displacement of each marker particle is shown as a line connecting positions of the same particle on Γ_0 and Γ_1. The fold classes in Fig. 2.1 are also modeled by the Lagrangian approach by varying the sign of $F_{\texttt{prop}}$ and $\psi_{\texttt{adv}}$ in Eq. (3.1) in agreement with Table 2.2.

The marker particle approach always works for the Class 2 fold with contribution from advection only when the characteristics are parallel. On the other hand, when $F_{\texttt{prop}} \neq 0$ the method does not resolve problems with characteristics that cross each other. Fig. 3.2(b) shows an example where characteristics cross and a so-called "swallow-tail" solution results in the core of a Class 1B fold. There exist Lagrangian methods (Sethian, 1985; Pons and Boissonnat, 2007) that can handle such situations in two dimensions, which resemble a viscosity solution of Eq. (2.23) as introduced in Crandall and Lions (1983), but in three dimensions the problem is much more complex. When the Lagrangian approach is used for initialization of the finite difference solver around the initial front Γ_0, the solution is only computed close to Γ_0 where the characteristics do not cross each other.

3.1.2 Initialization of the Finite Difference Solvers

A critical step in fast marching methods and other finite difference solvers is the initial computation of solution values at grid points in a narrow band around Γ_0, which enforces the boundary condition of the boundary value problem (2.23) (Algorithm 3.1, Step 1, 2 and 3). The fast marching algorithm propagates the solution downwind from Γ_0 to the whole solution domain in one pass. For each grid point that is being computed in Step 6 of Algorithm 3.1, only solution values of grid points tagged as *Known* on the upwind side are used. Therefore, the quality of the solution away from Γ_0 depends heavily on the quality of grid point values in the narrow band around Γ_0. Numerical errors introduced at this stage will propagate to the whole domain. This is also true for iterative solvers based on upwind finite differences. The initial narrow band around Γ_0 must be sufficiently wide, such that enough points are known on the upwind side when the fast marching method or the fast iterative method starts propagating the front away from Γ_0. A second order finite difference scheme requires a narrow band that is approximately twice as wide as a first order scheme. In the following we present a fast, stable and highly accurate solution to the initialization step for the general anisotropic case.

Let $\mathbf{x}_g$ be a grid point in the narrow band around Γ_0, and let $\mathbf{x}_i$ be the (unknown) point on Γ_0 where the characteristic through $\mathbf{x}_g$ intersects Γ_0 (Fig. 3.3). Note that the gradient direction, and thus the group velocity vector (2.24), is not known a priori in a point away from Γ_0. Assume local homogeneity with constant $F_{\texttt{prop}}$, $\psi_{\texttt{adv}}$ and $\mathbf{a}$, such that the group velocity vector in $\mathbf{x}_g$ is the same as that in $\mathbf{x}_i$. The group velocity is then given by Eq. (3.1), and the time of arrival $T(\mathbf{x}_g)$ is given by Eq. (3.2) with $t_0 = 0$.

In our implementation, Γ_0 is represented as a bicubic tensor product

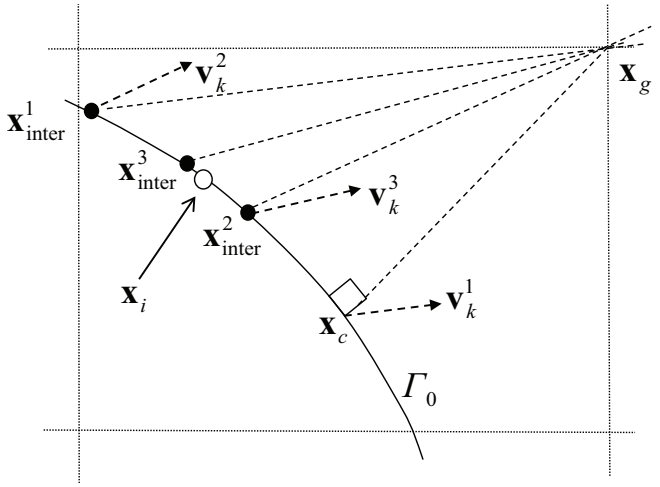


Fig. 3.3: Illustration for Algorithm 3.2 in two dimensions when Γ_0 is a curve. The small circle $\circ$ indicates the unknown intersection point between Γ_0 and the characteristic passing through grid point $\mathbf{x}_g$. The closest point found in Step 1 is $\mathbf{x}_c$. The markers $\bullet$ indicate intersection points from Step 3. Superscripts on $\mathbf{x}_{\texttt{inter}}$ and $\mathbf{v}_k$ indicate the iteration numbers in the loop over steps 2 and 3. Three iterations are shown.

Bézier surface in three dimensions, and as a cubic Bézier curve in two dimensions (Farin, 2002). Both are on parametric form with C^1 continuity, which is sufficient for $T(\mathbf{x})$ to be smooth near Γ_0 when the narrow band is initialized by the numerical scheme outlined below. Two fundamental operations were implemented to support Algorithm 3.2 in locating the intersection point $\mathbf{x}_i$,

(i) computing the closest point on Γ_0 from an arbitrary (sufficiently close) point, and

(ii) computing the intersection point between a straight line and Γ_0.

Both use Newton iteration schemes and techniques borrowed from computer aided design and computer graphics. When Γ_0 is a surface, the operations return a (u, v)-parameter tuple such that the closest point, or the intersection point, is found by evaluation of the parametric surface, $\mathbf{x} = \Gamma_0(u, v)$. Details on the Newton itertion schemes can be found in Sect. B.1 and B.2. Based on these two operations, we find an approximation for $\mathbf{x}_i$ in Eq. (3.2) by iteration as outlined in Algorithm 3.2 and illustrated in Fig. 3.3.

In Step 1 of the algorithm, the closest point $\mathbf{x}_c$ to $\mathbf{x}_g$ on Γ_0 is found as a first approximation to $\mathbf{x}_i$ with operation (i). Then in Step 2, the group

Algorithm 3.2 Computing the traveltime value $T(\mathbf{x}_g)$ in grid point $\mathbf{x}_g$ near Γ_0.

Intersection of characteristic through $\mathbf{x}_g$ with Γ_0 , cf. Fig. 3.3:

1. Find the closest point $\mathbf{x}_c$ on Γ_0 to grid point $\mathbf{x}_g$, and set

$$\mathbf{x}_k \leftarrow \mathbf{x}_c$$

2. Find the unit normal vector $\mathbf{n}_k$ of Γ_0 in $\mathbf{x}_k$ and compute the group velocity vector through $\mathbf{x}_k$ as

$$\mathbf{v}_k = F_{\mathtt{prop}}\,\mathbf{n}_k + \psi_{\mathtt{adv}}\,\mathbf{a}$$

3. Find the intersection point $\mathbf{x}_{\mathtt{inter}}$ between Γ_0 and the straight line through $\mathbf{x}_g$ with direction $\mathbf{v}_k$

4. If not converged, set

$$\mathbf{x}_k \leftarrow \mathbf{x}_{\mathtt{inter}}$$

go to Step 2

Computing $T(\mathbf{x}_g)$:

5. Set $\mathbf{x}_i \approx \mathbf{x}_{\mathtt{inter}}$ and find the normal vector $\mathbf{n}_i$ in $\mathbf{x}_i$.

6. Compute $T(\mathbf{x}_g)$ from Eq. (3.2) with $t_0 = 0$.

velocity vector $\mathbf{v}_k$ through $\mathbf{x}_c$ is found as a first approximation to the group velocity vector through $\mathbf{x}_i$. The first straight line found in Step 3, intersecting Γ_0 in $\mathbf{x}_{\mathtt{inter}}$ (operation (ii)), has the same direction as the group velocity vector through $\mathbf{x}_c$. Then the iteration in the loop over steps 2 and 3 moves $\mathbf{x}_{\mathtt{inter}}$ towards the exact intersection point, since the direction of $\mathbf{v}_k$ converges to the direction of the characteristic through $\mathbf{x}_g$. The iteration is stopped when $\mathbf{x}_{\mathtt{inter}}$ and $\mathbf{x}_k$ are sufficiently close. The deviation between these points can be measured geometrically by the distance $\|\mathbf{x}_{\mathtt{inter}} - \mathbf{x}_k\|$, or since $\Gamma_0(u, v)$ is a parametric surface, by differences in parameter values $|u_{\mathtt{inter}} - u_k|$ and $|v_{\mathtt{inter}} - v_k|$. Convergence is usually very fast, in most cases two or three iterations are sufficient.

In the isotropic case with $\psi_{\texttt{adv}} = 0$, for the parallel Class 1B fold, only the closest point computation in Step 1 is necessary, since the characteristic through $\mathbf{x}_g$ has the same direction as the vector $(\mathbf{x}_g - \mathbf{x}_c)$. Thus $\mathbf{x}_i = \mathbf{x}_c$ and the solution value is $T(\mathbf{x}_g) = \|\mathbf{x}_g - \mathbf{x}_c\|/F_{\texttt{prop}}$ under the same assumptions as above. When $F_{\texttt{prop}} = 0$, for the similar Class 2 fold, the direction of the characteristic is known a priori and equals the axial direction $\mathbf{a}$. Then $\mathbf{x}_i$ is found directly by operation (ii) as the intersection point between Γ_0 and the straight line through $\mathbf{x}_g$ with direction $\mathbf{a}$, and $T(\mathbf{x}_g) = \|\mathbf{x}_g - \mathbf{x}_i\|/\psi_{\texttt{adv}}$. And, in the special case when Γ_0 is (locally) planar, one iteration is sufficient.

Along with the traveltime value $T(\mathbf{x}_g)$, the gradient $\nabla T(\mathbf{x}_g)$ can be computed directly from $\mathbf{v}(\mathbf{x}_i) = \mathbf{v}_i$ and $\mathbf{n}(\mathbf{x}_i) = \mathbf{n}_i$, found upon convergence of Algorithm 3.2. The gradient direction in the intersection point $\mathbf{x}_i$ equals the direction of the surface normal $\mathbf{n}(\mathbf{x}_i)$. The assumption with local homogeneity implies that the gradient is constant along the characteristic between $\mathbf{x}_i$ and $\mathbf{x}_g$, thus $\nabla T(\mathbf{x}_g) = \|\nabla T(\mathbf{x}_i)\|\, \mathbf{n}(\mathbf{x}_i)$. The gradient is frequently referred to as the slowness vector in geometric optics since the elements of ∇T are the reciprocals of velocity. Thus, $\|\nabla T(\mathbf{x}_i)\|$ is the reciprocal of the speed $\mathbf{v}(\mathbf{x}_i) \cdot \mathbf{n}(\mathbf{x}_i)$ in the outward normal direction at $\mathbf{x}_i$. The gradient in $\mathbf{x}_g$ is then

$$\nabla T(\mathbf{x}_g) = \frac{1}{(\mathbf{v}(\mathbf{x}_i) \cdot \mathbf{n}(\mathbf{x}_i))}\, \mathbf{n}(\mathbf{x}_i).$$

3.1.3 Approximation of the Gradient with Upwind Finite Differences

The discretized version of Eq. (2.23) can be written

$$F_{\texttt{prop}} \|\nabla T_c\| + \psi_{\texttt{adv}}\, (\mathbf{a} \cdot \nabla T_c) = 1, \tag{3.3}$$

where T_c represents the unknown solution value in a grid point position $\mathbf{x}_c$ of the finite difference grid. We assume that $F_{\texttt{prop}}$, $\psi_{\texttt{adv}}$ and $\mathbf{a}$ are constant from the initial layer boundary Γ_0 to $\mathbf{x}_c$ along the characteristic. This equation must be solved locally, usually several times, for each grid point in the fast marching loop in Algorithm 3.1, or by other algorithms operating on the finite difference grid. The first step is then to find an approximation for the gradient ∇T_c in $\mathbf{x}_c$. We examine the two-dimensional case first.

Consider a triangle $\mathbf{x}_a\mathbf{x}_b\mathbf{x}_c$, where $\mathbf{x}_a$ and $\mathbf{x}_b$ are both upwind from $\mathbf{x}_c$ (Fig. 3.4). If solution values T_a and T_b are computed and T_c is unknown, and if the characteristic through $\mathbf{x}_c$ lies inside $\mathbf{x}_a\mathbf{x}_b\mathbf{x}_c$, with the triangle $\mathbf{x}_a\mathbf{x}_b\mathbf{x}_c$ being sufficiently small and acute, the solution value $T_c \geq \max(T_a, T_b)$ for the causality condition to hold. In this example, the $\mathbf{x}_a\mathbf{x}_c$ leg of the triangle is a diagonal

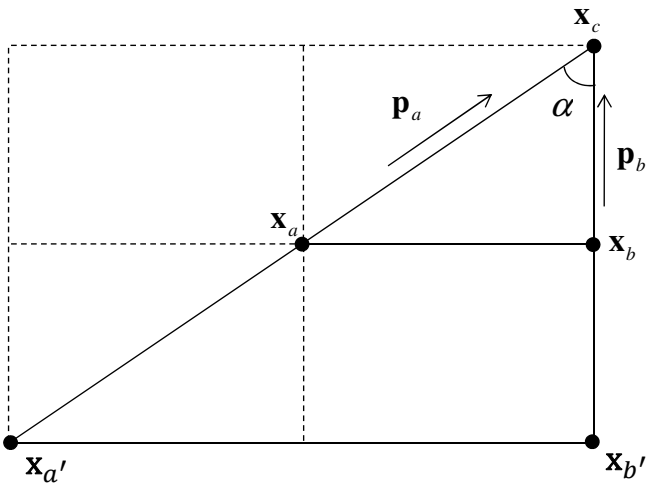


Fig. 3.4: The gradient ∇T_c in grid point position $\mathbf{x}_c$ is computed from known solution values T_a and T_b in $\mathbf{x}_a$ and $\mathbf{x}_b$, respectively, in the upwind direction from $\mathbf{x}_c$. The second order scheme also uses computed values $T_{a'}$ and $T_{b'}$ if they are smaller than, respectively, T_a and T_b.

of a grid cell. In addition to improving the directional resolution, the use of diagonal directions might be necessary in the anisotropic case since computation of the traveltime at $\mathbf{x}_c$ is not restricted to be based on grid points that are immediate neighbours (Sect. 3.1.5). If $\mathbf{x}_a$ and $\mathbf{x}_b$ are interior grid nodes that belong to the same rectangular grid cell as $\mathbf{x}_c$, there are also grid nodes in $\mathbf{x}'_a$ and $\mathbf{x}'_b$ in the upwind direction from $\mathbf{x}_c$ that can be used to construct a second order stencil. If we assume local homogeneity with constant $F_{\texttt{prop}}$, $\psi_{\texttt{adv}}$ and $\mathbf{a}$ near $\mathbf{x}_c$, the characteristics are straight lines such that the triangle $\mathbf{x}_{a'}\mathbf{x}_{b'}\mathbf{x}_c$ also contains the characteristic through $\mathbf{x}_c$.

Let $\mathbf{p}_a = (\mathbf{x}_c - \mathbf{x}_a)/\|\mathbf{x}_c - \mathbf{x}_a\|$ and $\mathbf{p}_b = (\mathbf{x}_c - \mathbf{x}_b)/\|\mathbf{x}_c - \mathbf{x}_b\|$ be downwind directions of the two triangle sides meeting at $\mathbf{x}_c$, and let the 2×2 matrix P have rows $\mathbf{p}_a^T$ and $\mathbf{p}_b^T$, i.e.

$$P = \begin{pmatrix} \mathbf{p}_a^T \\ \mathbf{p}_b^T \end{pmatrix}. \tag{3.4}$$

If ∇T_c is an approximation for the gradient at $\mathbf{x}_c$, then $v_a = \nabla T_c \cdot \mathbf{p}_a$ and $v_b = \nabla T_c \cdot \mathbf{p}_b$ are approximations to the directional derivatives of T along $\mathbf{p}_a$ and $\mathbf{p}_b$, respectively. This can be written in matrix form as

$$\mathbf{v} = \begin{pmatrix} v_a \\ v_b \end{pmatrix} = P\,\nabla T_c.$$

Clearly P is non-singular since $\mathbf{p}_a$ and $\mathbf{p}_b$ are linear independent, and thus

$$\nabla T_c = P^{-1}\mathbf{v}.$$

Using backward differences, the first order and second order approximations for the directional derivatives can be written (second order in parentheses)

$$\begin{aligned} v_a &= \frac{T_c - T_a}{\|\mathbf{x}_c - \mathbf{x}_a\|} \qquad \left(\frac{3T_c - 4T_a + T_{a'}}{2\|\mathbf{x}_c - \mathbf{x}_a\|}\right), \\ v_b &= \frac{T_c - T_b}{\|\mathbf{x}_c - \mathbf{x}_b\|} \qquad \left(\frac{3T_c - 4T_b + T_{b'}}{2\|\mathbf{x}_c - \mathbf{x}_b\|}\right). \end{aligned}$$

More generally, the difference approximations can be expressed as linear combinations of the solution values as

$$\begin{aligned} v_a &= r_a T_c + s_a \\ v_b &= r_b T_c + s_b, \end{aligned}$$

using first order and second order difference operators (second order in parenthesis)

$$\begin{aligned} r_a &= \frac{1}{\|\mathbf{x}_c - \mathbf{x}_a\|} \quad \left(\frac{3}{2\|\mathbf{x}_c - \mathbf{x}_a\|}\right), \quad s_a = \frac{-T_a}{\|\mathbf{x}_c - \mathbf{x}_a\|} \quad \left(\frac{-4T_a + T_{a'}}{2\|\mathbf{x}_c - \mathbf{x}_a\|}\right), \\ r_b &= \frac{1}{\|\mathbf{x}_c - \mathbf{x}_b\|} \quad \left(\frac{3}{2\|\mathbf{x}_c - \mathbf{x}_b\|}\right), \quad s_b = \frac{-T_b}{\|\mathbf{x}_c - \mathbf{x}_b\|} \quad \left(\frac{-4T_b + T_{b'}}{2\|\mathbf{x}_c - \mathbf{x}_b\|}\right). \end{aligned}$$

The directional derivatives can be written

$$\mathbf{v} = \mathbf{r}T_c + \mathbf{s}, \tag{3.5}$$

where $\mathbf{r} = (r_a, r_b)^T$ and $\mathbf{s} = (s_a, s_b)^T$. The gradient then becomes

$$\nabla T_c = P^{-1}(\mathbf{r}T_c + \mathbf{s}). \tag{3.6}$$

In an algorithmic setting one will typically switch between first order and second order difference operators to ensure a proper upwind construction that satisfies the causality condition. For example, the second order backward operator for v_a above requires $T_{a'} < T_a$. If this is not true, the first order approximation is used instead. For an overview of the the finite difference operators, see Appendix C.

3.1.4 Local Solver over Triangles and Tetrahedra

The gradient ∇T_c depends linearly on T_c and other grid point values in the upwind direction from $\mathbf{x}_c$, both for the first order and second order operator,

but the first term of Eq. (3.3) is non-linear. Separating non-linear and linear terms in Eq. (3.3) and squaring the equation, we get

$$F_{\mathtt{prop}}^2 \|\nabla T_c\|^2 = \left(1 - \psi_{\mathtt{adv}}\, \mathbf{a} \cdot \nabla T_c\right)^2 . \tag{3.7}$$

Since $\|\nabla T_c\|^2 = (\nabla T_c)^T \nabla T_c$, the left hand side can be written as the quadratic expression

$$F_{\mathtt{prop}}^2 \mathbf{v}^T (PP^T)^{-1} \mathbf{v},$$

where $\mathbf{v}$ is given by Eq. (3.5), and P by Eq. (3.4). Inserting for $\mathbf{v}$, the left hand side becomes

$$F_{\mathtt{prop}}^2 \left[\left(\mathbf{r}^T Q \mathbf{r}\right) T_c^2 + \left(2\mathbf{r}^T Q \mathbf{s}\right) T_c + \mathbf{s}^T Q \mathbf{s} \right],$$

where $Q = \left(PP^T\right)^{-1}$. The right hand side of Eq. (3.7) can be written

$$\psi_{\mathtt{adv}}^2 \gamma_r^2 T_c^2 + (2\psi_{\mathtt{adv}}^2 \gamma_r \gamma_s - 2\psi_{\mathtt{adv}} \gamma_r) T_c + \psi_{\mathtt{adv}}^2 \gamma_s^2 - 2\psi_{\mathtt{adv}} \gamma_s + 1,$$

where

$$\begin{aligned} \gamma_r &= \mathbf{a}^T P^{-1} \mathbf{r} \quad \text{and} \\ \gamma_s &= \mathbf{a}^T P^{-1} \mathbf{s}. \end{aligned} \tag{3.8}$$

Merging these expressions and rearranging, we obtain a quadratic polynomial in T_c,

$$(A_{\mathtt{prop}} + A_{\mathtt{adv}}) T_c^2 + (B_{\mathtt{prop}} + B_{\mathtt{adv}}) T_c + (C_{\mathtt{prop}} + C_{\mathtt{adv}}) - 1 = 0, \tag{3.9}$$

where coefficients with contribution from normal propagation and advection have been separated as

$$\begin{aligned} A_{\mathtt{prop}} &= F_{\mathtt{prop}}^2 \left(\mathbf{r}^T Q \mathbf{r}\right), & A_{\mathtt{adv}} &= -\psi_{\mathtt{adv}}^2 \gamma_r^2, \\ B_{\mathtt{prop}} &= F_{\mathtt{prop}}^2 \left(2\mathbf{r}^T Q \mathbf{s}\right), & B_{\mathtt{adv}} &= 2\psi_{\mathtt{adv}} \gamma_r - 2\psi_{\mathtt{adv}}^2 \gamma_r \gamma_s, \\ C_{\mathtt{prop}} &= F_{\mathtt{prop}}^2 \left(\mathbf{s}^T Q \mathbf{s}\right), & C_{\mathtt{adv}} &= 2\psi_{\mathtt{adv}} \gamma_s - \psi_{\mathtt{adv}}^2 \gamma_s^2. \end{aligned}$$

Clearly, the particular speed profile used in our front propagation problem results in an analytical expression for the local solver that can be evaluated directly to find the solution value in a grid point. This might not be true for anisotropic problems with more complex speed profiles that must be solved iteratively (Vladimirsky, 2001). Since a quadratic equation must be solved to find T_c, there are two distinct roots in general, and we have to choose the one that satisfies the causality condition.

In the special case of the Class 1B fold (parallel) with $\psi_{\mathtt{adv}} = 0$, which corresponds to isotropic front propagation, the quadratic equation becomes

$$\left(\mathbf{r}^T Q \mathbf{r}\right) T_c^2 + \left(2\mathbf{r}^T Q \mathbf{s}\right) T_c + \mathbf{s}^T Q \mathbf{s} = 1/F_{\mathtt{prop}}^2. \tag{3.10}$$

We note that

$$Q = (PP^T)^{-1} = \frac{1}{\sin^2\alpha}\begin{pmatrix} 1 & -\cos\alpha \\ -\cos\alpha & 1 \end{pmatrix},$$

where α is the interior angle of the triangle at $\mathbf{x}_c$ (Fig. 3.4). If we solve the isotropic case over a rectangular grid without using diagonal directions in the stencil construction, then $\alpha = 90$ degrees and Q is the identity matrix. Using a first order scheme, Eq. (3.10) then reverts to

$$\left(\frac{T_c - T_a}{\Delta x}\right)^2 + \left(\frac{T_c - T_b}{\Delta y}\right)^2 = \frac{1}{F_{\mathtt{prop}}^2}. \tag{3.11}$$

This is the discretized version of Eq. (2.4) which agrees with the first order standard Godunov upwind scheme for solving the eikonal equation (Sethian, 1999b).

The other special case, with $F_{\mathtt{prop}} = 0$, models Class 2 (similar) folds. Then the discriminant of Eq. (3.9) is zero and the equation is linear. Solving for T_c we get

$$T_c = \frac{-B_{\mathtt{adv}}}{2A_{\mathtt{adv}}} = \frac{1 - \psi_{\mathtt{adv}}\gamma_s}{\psi_{\mathtt{adv}}\gamma_r} = \frac{1 - \psi_{\mathtt{adv}}\,\mathbf{a}^T P^{-1}\mathbf{s}}{\psi_{\mathtt{adv}}\,\mathbf{a}^T P^{-1}\mathbf{r}}, \tag{3.12}$$

which is the discretized version of Eq. (2.11). The downwind direction, and equivalently, the direction of the characteristics represented by the ODE (2.24), now reduces to $d\mathbf{x}/dt = \psi_{\mathtt{adv}}(\mathbf{x})\,\mathbf{a}$. Thus, the downwind direction is uniquely defined by the axial direction $\mathbf{a}$ of the fold. If $\mathbf{a}$ is constant, Eq. (3.12) could then be recast to a graph in a Cartesian system with abscissa orthogonal to $\mathbf{a}$, and thus solved explicitly without a finite difference formulation.

Conceptually, extension of the numerical scheme to three dimensions is straightforward. The stencil is then spanned by a tetrahedron with four vertices, say $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$ and $\mathbf{x}_c$, where $\mathbf{x}_c$ is downwind from the three other vertices. The tetrahedron must embed the characteristic through $\mathbf{x}_c$ such that the three-dimensional group velocity vector (2.24) points away from the tetrahedron. Assume that solution values T_1, T_2 and T_3 are known and that T_c is unknown. We have the three downwind directions along the legs of the tetrahedron meeting at vertex $\mathbf{x}_c$, namely $\mathbf{p}_1 = (\mathbf{x}_c - \mathbf{x}_1)/\|\mathbf{x}_c - \mathbf{x}_1\|$, $\mathbf{p}_2 = (\mathbf{x}_c - \mathbf{x}_2)/\|\mathbf{x}_c - \mathbf{x}_2\|$, and $\mathbf{p}_3 = (\mathbf{x}_c - \mathbf{x}_3)/\|\mathbf{x}_c - \mathbf{x}_3\|$. These unit vectors constitute the rows of the 3×3 matrix P. The approximation to the gradient takes the same form as Eq. (3.6), with $\mathbf{r}$ and $\mathbf{s}$ now being vectors in three dimensions. The general discretized PDE (3.9) also takes the same form with the axial direction $\mathbf{a}$ being a vector in three dimensions.

The fast marching method and most other single-pass methods rely on the Hamiltonian of Eq. (2.23) being convex in the gradient argument $\mathbf{p} = \nabla T$ (Vladimirsky, 2001). Since the second term of $H(\mathbf{x}, \mathbf{p})$ is linear in $\mathbf{p}$, we only need to examine the first term, $F_{\texttt{prop}}(\mathbf{x})\|\nabla T(\mathbf{x})\|$. The Euclidean norm is linear and strictly increasing radially away from the origin $\mathbf{p} = \mathbf{0}$, and thus convex. $H(\mathbf{x}, \mathbf{p})$ is then convex downward for $F_{\texttt{prop}} > 0$, when modeling Class 1A, 1B and 1C folds with convergent dip isogons, and convex upward for $F_{\texttt{prop}} < 0$, when modeling Class 3 folds with divergent dip isogons. A systematic approach for checking causality and choosing the correct root that satisfies the causality condition can be found in Qian *et al.* (2007).

3.1.5 Dynamic Upwind Stencil Construction for Anisotropic Problems

The original fast marching method (FMM) by Sethian (1996) hinges on the important property of isotropic problems that the gradient ∇T points in the same direction as the characteristic of the PDE everywhere in the solution domain. For anisotropic problems, when modeling fold classes other than Class 1B, this is no longer true. While the fast marching method only uses adjacent neighbours when updating a grid point, the Ordered Upwind Method for anisotropic problems (Sethian and Vladimirsky, 2003) considers a larger neighbourhood. Since this neighbourhood is bounded by a so-called anisotropy coefficient, a one-pass method similar to fast marching is still achieved, but with the computational complexity of $O(\Upsilon N \log N)$, where $\Upsilon \geq 1$ is the anisotropy coefficient. Since the neighbourhood can be quite large, and the number of re-computations of trial points can be large, the algorithm is more time consuming than FMM. The implementation is also more complex. A simplified scheme was proposed in Lin (2003). This method uses only immediate neighbours when updating a grid point, similarly to FMM. Unlike the Ordered Upwind Method, the simplified scheme is not guaranteed to converge to the unique viscosity solution of the PDE.

The scheme we propose here is inspired by both these methods. The finite difference grid still has a rectangular pattern, but we use diagonal directions in the stencil construction both for better directional resolution and to guarantee the causality condition. The neighbourhood used for updating a grid point is tentatively larger than in FMM, but the basic structure and the computational complexity of FMM are maintained. Like in Lin (2003), we have no mathematical proof of convergence to the viscosity solution, but experiments show that the accuracy is sufficient for the applications of fold modeling that we target (Petersen *et al.*, 2007; Petersen and Hjelle, 2008). The larger neigh-

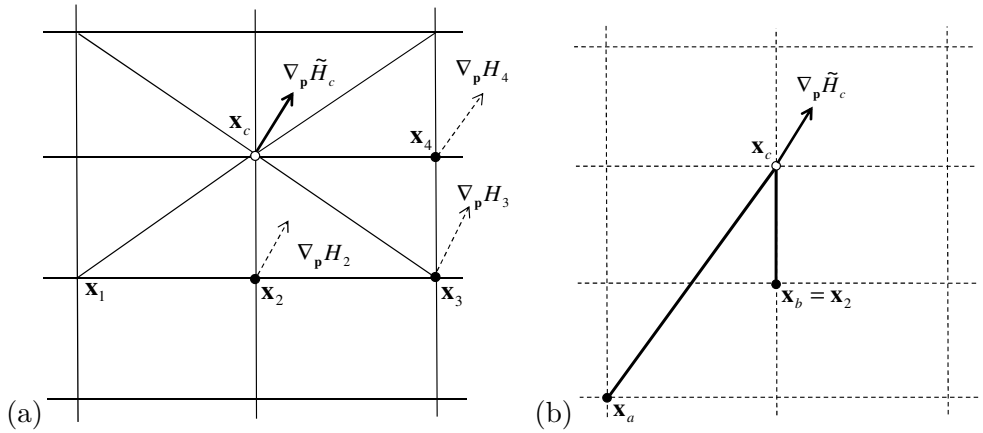


Fig. 3.5: (a) A situation in two dimensions when the point in $\mathbf{x}_c$ is being computed. Three immediate neighbours marked with • have already been computed. An approximation $\nabla_{\mathbf{p}}\widetilde{H}_c$ to the group velocity in $\mathbf{x}_c$ is based on $\nabla_{\mathbf{p}}H_2$ and $\nabla_{\mathbf{p}}H_3$. (b) A stencil is constructed from points $\mathbf{x}_a$, $\mathbf{x}_b$ and $\mathbf{x}_c$ such that $\nabla_{\mathbf{p}}\widetilde{H}_c$ points away from the triangle $\mathbf{x}_a\mathbf{x}_b\mathbf{x}_c$.

bourhood used in the dynamic stencil construction proposed here (Fig. 3.5) ensures that information always flows in the downwind direction, like in FMM. To compute $T(\mathbf{x}_c)$ by solving Eq. (3.9), we want to find a simplex that embeds the characteristic through $\mathbf{x}_c$ such that a stencil can be constructed that satisfies the causality condition (Fig. 3.5). Ideally, we should first find the group velocity vector through $\mathbf{x}_c$ as

$$\nabla_{\mathbf{p}}H_c = F_{\mathtt{prop}}\frac{\nabla T(\mathbf{x}_c)}{\|\nabla T(\mathbf{x}_c)\|} + \psi_{\mathtt{adv}}\,\mathbf{a}, \tag{3.13}$$

which has the same direction as the characteristic through $\mathbf{x}_c$. Since the gradient $\nabla T(\mathbf{x}_c)$ is unknown until $T(\mathbf{x}_c)$ is computed, we must find an approximation $\nabla_{\mathbf{p}}\widetilde{H}_c \approx \nabla_{\mathbf{p}}H_c$ based on points on the upwind side that are already known. Fig. 3.5(a) illustrates a situation in two dimensions. On the upwind side from $\mathbf{x}_c$, there are points that have been computed and marked either as known or trial points. The points on the downwind side are all unknown. Only known points that are immediate neighbours to $\mathbf{x}_c$ will be considered when estimating $\nabla_{\mathbf{p}}\widetilde{H}_c$. Immediate neighbours means that they are neighbours to $\mathbf{x}_c$ horizontally, vertically or diagonally across a grid cell, as the points $\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{x}_3$ and $\mathbf{x}_4$ in the figure. In the two-dimensional case, experiments show that in most cases there are exactly two such known points on the upwind side, in other cases there are up to five, and in some rare cases there is just one known point on the upwind side among the immediate neighbours. The gradient in each known point $\mathbf{x}_i$ can be consistently computed by Eq. (3.6) with upwind

finite differences on the same stencil that was used by the local solver when T_i was computed. The direction of the characteristic through $\mathbf{x}_i$ is then given by Eq. (2.24). Then $\nabla_{\mathbf{p}}\widetilde{H}_c$ is determined by one of these two rules, depending on the configuration of known points on the upwind side.

(i) If there are two or more known points among which there are pairwise neighbours, $\nabla_{\mathbf{p}}\widetilde{H}_c$ is computed as the mean of the group velocity vectors through the most correlated pair of neighbours. The rationale behind this approach is to avoid outliers and reduce noise. The degree of correlation is measured by the angle between the vectors.

(ii) If there are no pairwise neighbours, $\nabla_{\mathbf{p}}\widetilde{H}_c$ is set as the mean of the group velocity vectors through the known points among the immediate neighbours.

In Fig. 3.5(a), rule (i) is applied to compute $\nabla_{\mathbf{p}}\widetilde{H}_c$ from the group velocity vectors in $\mathbf{x}_2$ and $\mathbf{x}_3$. Next, a simplex with known points on the upwind side is sought for that embeds the characteristic through $\mathbf{x}_c$, that is, such that the group velocity in $\nabla_{\mathbf{p}}\widetilde{H}_c$ points from the simplex. This is not always possible by using only immediate neighbours. For example, in Fig. 3.5(a), $\nabla_{\mathbf{p}}\widetilde{H}_c$ points from the simplex $(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_c)$, but the point in $\mathbf{x}_1$ has not yet been computed, and the simplex is thus not a candidate for the stencil. On the other hand, by searching in the upwind direction, a known point is found on the next grid line such that a valid stencil can be constructed from the simplex $(\mathbf{x}_a, \mathbf{x}_b, \mathbf{x}_c)$ in Fig. 3.5(b). It may happen that $\mathbf{x}_1$ never will be computed, for example, if it is near the boundary of the domain. Then a void area with non-computed grid points may be left in the downwind direction along the characteristic through $\mathbf{x}_1$. Examples are shown in the lower left and upper right corner of Fig. 3.6, and in the lower left corner of Fig. 3.7(b). When this scheme is inserted into Step 6 of Algorithm 3.1, the same $O(N \log N_h)$ computational complexity of fast marching is retained.

The solution of Hamilton-Jacobi equations may develop corners and cusps with singularities in the derivatives even when the initial conditions are smooth. In these locations the solution does not satisfy the equation in the classical sense. The upwind stencil construction described above, and the ordering of how points are accepted in Algorithm 3.1, Step 4, resolves this by producing a viscosity solution (Crandall and Lions, 1983; Crandall *et al.*, 1984) similar to how this is achieved by the original fast marching algorithm for isotropic front propagation. Examples can be seen in the core of the folds in Fig. 3.7(b).

3.2 Numerical Examples

Fig. 3.6 shows an example of using Algorithm 3.1 and the numerical schemes above to simulate a Class 3 fold. The transverse lines representing dip isogons, or equivalently characteristics of Eq. (2.23), has been computed directly from Eq. (3.13) in each grid point along with $T(\mathbf{x}_c)$. An example in three dimensions of a Class 1B fold is shown in Fig. 2.4 on page 21.

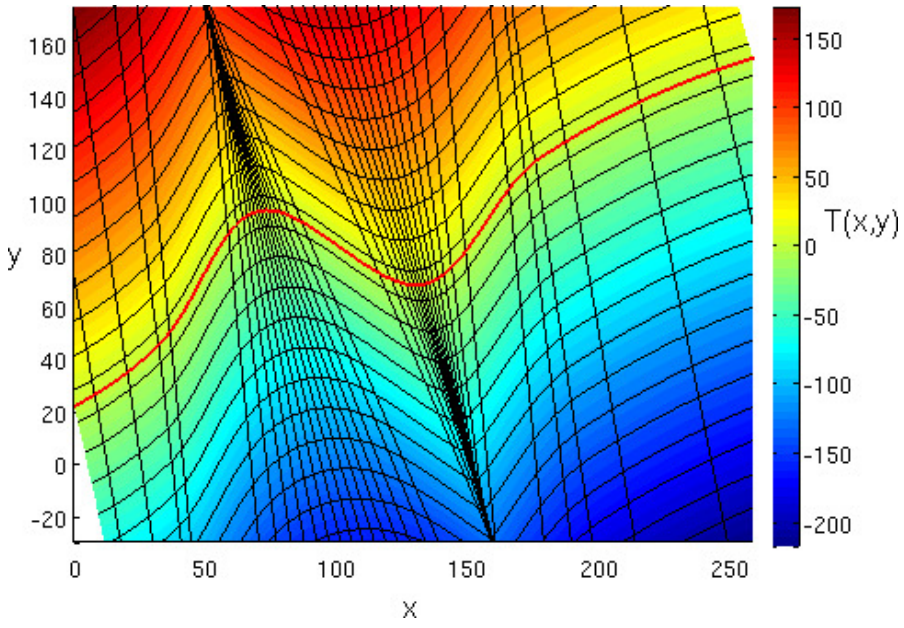
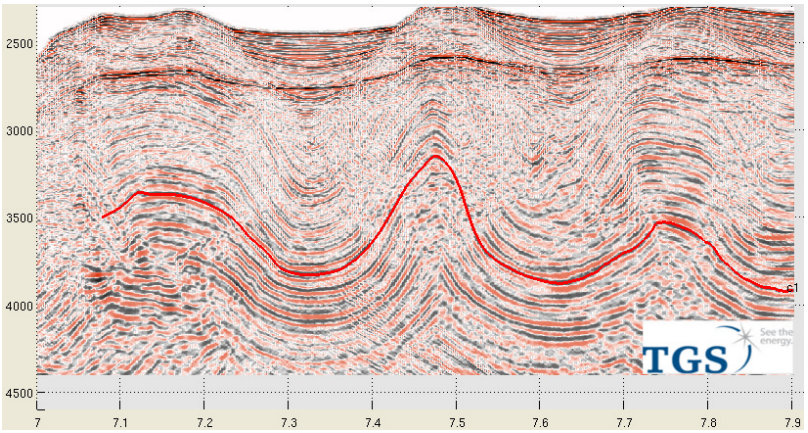


Fig. 3.6: Class 3 fold style represented as a time of arrival field with $F_{\texttt{prop}} = -0.2$, and axial direction $\psi_{\texttt{adv}}\,\mathbf{a} = (-0.25, 1.0)$. Isocurves of the T-field are shown as field lines propagating from the red reference curve Γ_0. A set of (divergent) dip isogons is also shown as transverse lines emanating from Γ_0, corresponding to characteristics of Eq. (2.23).

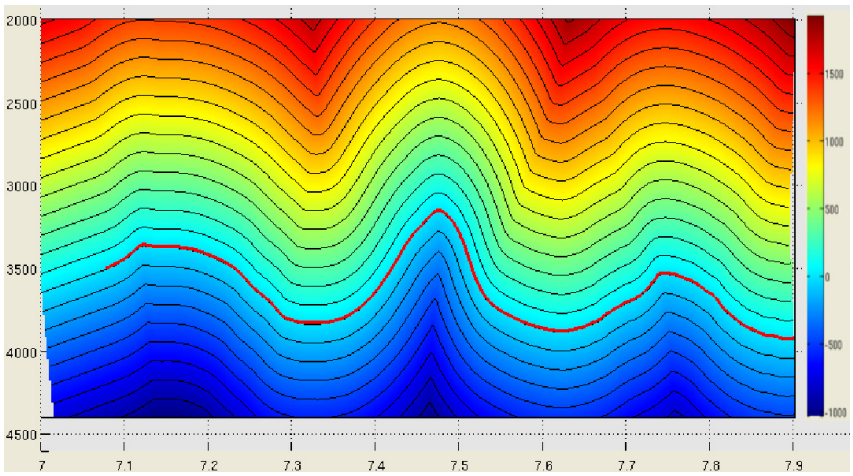
The numerical framework presented in this chapter, together with extensions presented in the next chapter, serve as the numerical core in the dR/mR-workflows in CES for process based data restoration and model reconstruction. Fold modeling is implemented as a separate process that can be preceded or proceeded by other structural processes, such as faulting presented in Sect. 4.4.4. The scheme extends to three dimensions, where the interfaces are surfaces (Petersen *et al.*, 2007). Fig. 3.7 shows the CES result from modeling compressional folds from a seismic line across the Karama Block in the North Makassar Strait outside Sulawesi. An interface Γ_0 is digitized from the

seismic line in (a) and the T-field shown in (b) is computed. The folds are of Class 1C type with small $\psi_{\texttt{adv}}/F_{\texttt{prop}}$ ratio that make them almost a Class 1B type. Properties representing velocity and density are then distributed from Γ_0 in the directions of the characteristics of the underlying PDE. The image in (c) shows the corresponding migrated seismic response. Note that only one interface is used for modeling the folds in this example. This is not always sufficient since the geometry of a fold may vary through a volume. In many cases layers delimited by two or more interfaces must be modeled to produce one T-field that is a blending between two or more T-fields. This is explained in detail in Sect. 4.3, where the method for populating properties in space is also explained. Other numerical examples with plots of a gradient field and T-fields in two and three dimensions of all the five fold classes can be found in Chapter 2.

(a)



(b)



(c)

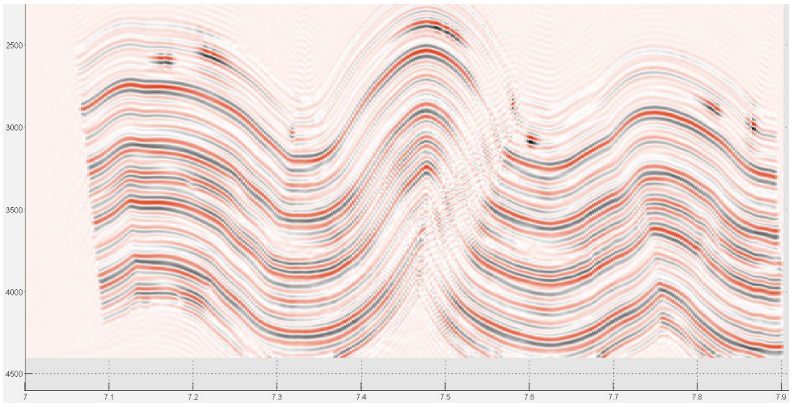


Fig. 3.7: (a) Seismic line with compressional folds across the Karama Block in the North Makassar Strait outside Sulawesi (courtesy of TGS), with a digitized interface in red. (b) The T-field corresponding to the digitized interface. (c) Migrated seismic response derived from 1D density and velocity distributions over the simulated fold in (b).

Chapter 4

A Numerical Framework to Support Data Restoration and Model Reconstruction

With Extensions of the Hamilton-Jacobi Framework

This chapter extends the mathematical and numerical frameworks from Chapters 2 and 3 to support processes for data restoration, model reconstruction and property distribution. The techniques we develop rely on a parameterization of the 3D space that serves as a reference system for both property distribution and spatial transformations involving deformations. The parameterization also forms the basis of a numerical framework for computing metric tensors and assessing strain. Contrary to other published methods for restoration, the methods presented here are meshless without the need for constructing a complex tetrahedral mesh to honor surfaces of a structural model.

The research presented in this chapter was motivated by the concept of Earth Recursion and the dR/mR-workflows outlined in Chapter 1, but it should be emphasized that the techniques we develop have applicability beyond the dR/mR-workflows. The results have been implemented in CES, and numerical examples from CES are given. Note that the main focus is on the mathematical and the numerical frameworks that can lay the foundation for implementation of processes for restoration and model reconstruction. Details of the actual implementation are beyond the scope of this thesis and will be published elsewhere.

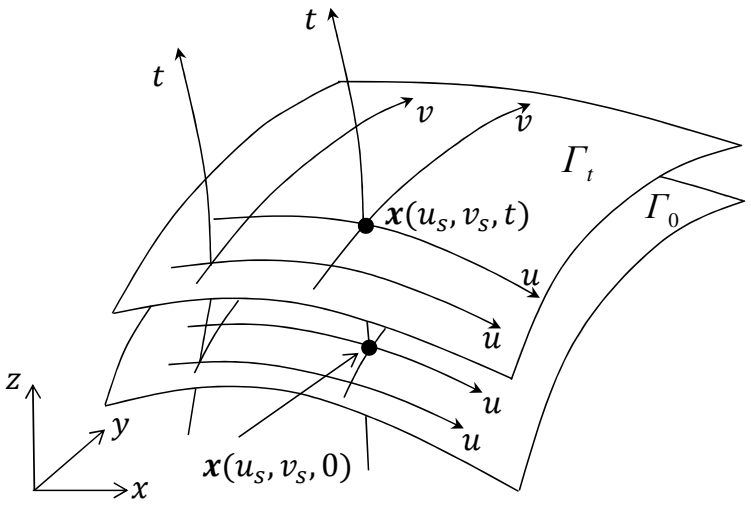


Fig. 4.1: Parameterization of the 3D space. Two isosurfaces Γ_0 and Γ_t of the T-field are shown with two u-lines and two v-lines of the curvilinear grid on each isosurface. In addition, two t-lines are shown.

4.1 Parameterization of the 3D Space

We want to impose a parameterization on the 3D space such that any point $\mathbf{x} = (x, y, z)$ has a parametric representation

$$\mathbf{x}(u, v, t) = (\, x(u, v, t),\, y(u, v, t),\, z(u, v, t)\,).$$

Then we have a unique mapping from the parameter space to the Euclidean space by

$$\mathbf{u} = (u, v, t) \longrightarrow \mathbf{x} = (x, y, z), \tag{4.1}$$

and the inverse mapping

$$\mathbf{x} = (x, y, z) \longrightarrow \mathbf{u} = (u, v, t). \tag{4.2}$$

This parameterization can be associated with a curvilinear and logically rectangular grid in the Euclidean space consisting of directed u-lines, v-lines and t-lines as illustrated in Fig. 4.1. We assume that the three parameter directions obey the right-hand rule as in the figure. Along each parameter line, the respective parameter value is monotonically increasing/decreasing while the two other parameter values are constant. Thus, the intersection curve between an isosurface with constant u-vale and an isosurface with constant v-value is a t-line. Similarly, a v-isosurface and a t-isosurface intersect in a u-line, and a t-isosurface and a u-isosurface intersect in a v-line. This implies the following orthogonality properties.

P1: the normal vector of a u-isosurface passing through a point $\mathbf{x}$ is orthogonal to the t-line through $\mathbf{x}$,

P2: the normal vector of a v-isosurface passing through a point $\mathbf{x}$ is orthogonal to the t-line through $\mathbf{x}$.

A natural choice for the field $\{t(\mathbf{x})\}$ of the parameterization is to use the time of arrival field T in which the isosurfaces represent a continuum of layer boundaries of a fold (Sect. 2.3). The T-field is known from the numerical solution of Eq. (2.23) on the computational grid, where parameters $F_{\texttt{prop}}$, $\psi_{\texttt{adv}}$ and $\mathbf{a}$ are properly chosen from Table 2.2 to model the actual fold type. Recall that the reference surface $\Gamma_0(u, v)$ in the numerical framework for computing the T-field was represented as a parametric surface, and that $\Gamma_0(u, v)$ defined the zero set of T (Sect. 3.1.2). We adopt this parameterization for points in 3D space located on the surface Γ_0, which implies

$$\mathbf{x}(u, v, 0) = \Gamma_0(u, v).$$

If the fields $\{u(\mathbf{x})\}$ and $\{v(\mathbf{x})\}$ were known in the whole domain, then any isosurface of the T-field, which corresponds to a layer boundary, could be associated with an implicitly defined parametric surface $\Gamma_t(u, v)$ such that

$$\mathbf{x}(u, v, t) = \Gamma_t(u, v).$$

We seek a meshless representation of the parameterization on the discrete finite difference grid where the T-field is computed. Thus, we want to find a set of parameter triples

$$\{(u, v, t)_{i,j,k}\},\ i \in \mathcal{I}_{\mathbf{x}},\ j \in \mathcal{I}_{\mathbf{y}},\ k \in \mathcal{I}_{\mathbf{z}}$$

over the $m \times n \times p$ grid such that the mapping in Eq. (4.2) can be performed locally by a trivariate interpolator. Here we have introduced the index sets

$$\mathcal{I}_{\mathbf{x}} = \{1, \ldots, m\}, \quad \mathcal{I}_{\mathbf{y}} = \{1, \ldots, n\} \quad \text{and} \quad \mathcal{I}_{\mathbf{z}} = \{1, \ldots, p\} \tag{4.3}$$

that we will use throughout this chapter to define scalar fields on the computational grid. The grid values $\{t_{i,j,k}\}$ can then be taken directly from the computed T-field. The corresponding scalar fields for u and v that we derive in the next section are

$$S_u = \{u_{i,j,k}\},\ i \in \mathcal{I}_{\mathbf{x}},\ j \in \mathcal{I}_{\mathbf{y}},\ k \in \mathcal{I}_{\mathbf{z}} \quad \text{and}$$
$$S_v = \{v_{i,j,k}\},\ i \in \mathcal{I}_{\mathbf{x}},\ j \in \mathcal{I}_{\mathbf{y}},\ k \in \mathcal{I}_{\mathbf{z}}.$$

We will call S_u and S_v parameter distribution fields, or S-fields for short. Since the scalar fields T, S_u, and S_v are defined on the same regular grid,

the mapping in Eq. (4.2) can be done by simple table look-up for a grid point $\mathbf{x}((u,v,t)_{i,j,k})$, and by a local trivariate interpolator for an arbitrary point $\mathbf{x}(u,v,t)$. To perform the inverse operation in Eq. (4.1), we need to construct functions $(u,v,t) \to x$, $(u,v,t) \to y$ and $(u,v,t) \to z$. We return to this later.

Before we can derive the S-fields, we need to constrain the direction of the u, v and t-lines. When the T-field models a fold, it is tempting to align the t-lines with the direction of the dip isogons. Recall from Sect. 2.4.1 that dip isogons coincide with the characteristic curves of the Hamilton-Jacobi equation that models the T-field. Thus, they point in the direction of the group velocity vector which is uniquely given by

$$\nabla_{\mathbf{p}} H(\mathbf{x},\mathbf{p}) = F_{\mathtt{prop}}(\mathbf{x}) \frac{\nabla T(\mathbf{x})}{\|\nabla T(\mathbf{x})\|} + \psi_{\mathtt{adv}}(\mathbf{x})\,\mathbf{a}. \tag{4.4}$$

We could make this more general by replacing $F_{\mathtt{prop}}$ and $\psi_{\mathtt{adv}}$ with other functions such that the direction of t-lines is any linear combination of the gradient direction $\nabla T(\mathbf{x})/\|\nabla T(\mathbf{x})\|$ and the axial direction $\mathbf{a}$. Or, we could restrict the direction to the gradient direction such that t-lines are always orthogonal to isosurfaces Γ_t of T. For notational clarity when deriving the S-fields, and for reusing the numerical framework developed in Chapter 3, we choose to align t-lines with $\nabla_{\mathbf{p}} H$. We will also see that this has physical relevance by looking at the physical processes and strain states set up during fold formation.

4.2 Construction of the S-Fields

In the narrowband around the reference surface $\Gamma_0(u,v)$, grid point values for S_u and S_v can be found by using the same scheme as that used for initialization of T-values in Sect. 3.1.2. Assume local homogeneity such that $F_{\mathtt{prop}}$, $\psi_{\mathtt{adv}}$ and $\mathbf{a}$ are constant in the narrowband arround $\Gamma_0(u,v)$, and that the t-lines are aligned with the characteristics with direction $\nabla_{\mathbf{p}} H$. Under these assumptions, for a grid point location $\mathbf{x}_g$ near $\Gamma_0(u,v)$, $S_u(\mathbf{x}_g)$ and $S_v(\mathbf{x}_g)$ should take the parameter values u and v in the parameter space of $\Gamma_0(u,v)$ where a straight line with direction $\nabla_{\mathbf{p}} H_g$ through $\mathbf{x}_g$ intersects $\Gamma_0(u,v)$, as shown in Fig. 4.2. Then, grid values $S_u(\mathbf{x}_g) = u_{\mathtt{inter}}$ and $S_v(\mathbf{x}_g) = v_{\mathtt{inter}}$, where $(u_{\mathtt{inter}}, v_{\mathtt{inter}})$ is the parameter pair of the last intersection point found upon convergence of Algorithm 3.2. Away from $\Gamma_0(u,v)$, where the T-field is computed by fast marching or another upwind finite differences solver, the S-field can be computed together with the T-values as shown in the following discussion.

Recall the orthogonality properties P1 and P2 from the previous section. The normal vector of a u-isosurface has the direction ∇S_u, and the normal

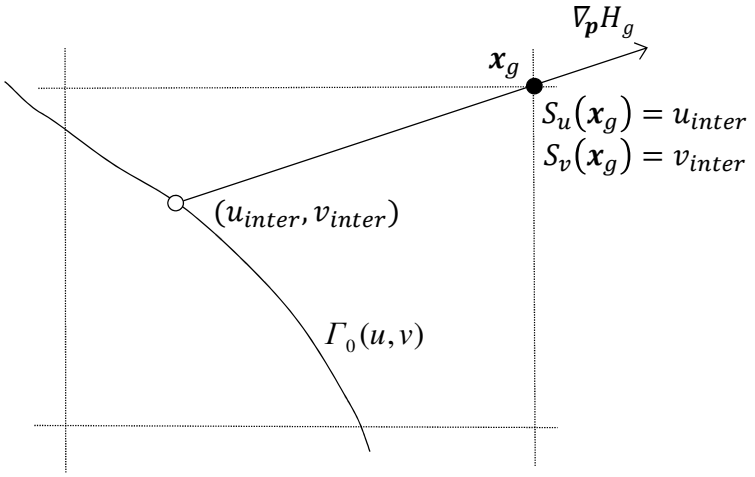


Fig. 4.2: Computation of $S_u(\mathbf{x}_g)$ and $S_v(\mathbf{x}_g)$ in a grid point $\mathbf{x}_g$ near the reference surface $\Gamma_0(u, v)$.

vector of a v-isosurface has the direction ∇S_v. Then, since we chose to align the t-lines with $\nabla_{\mathbf{p}} H$, we conclude that the S-fields must obey the orthogonality conditions

$$\begin{aligned} \nabla S_u(\mathbf{x}) \cdot \nabla_{\mathbf{p}} H(\mathbf{x}, \mathbf{p}) &= 0 \quad \text{and} \\ \nabla S_v(\mathbf{x}) \cdot \nabla_{\mathbf{p}} H(\mathbf{x}, \mathbf{p}) &= 0. \end{aligned} \tag{4.5}$$

These two equations are independent and can be discretized separately on the finite difference grid. Recall from Sect. 3.1.3 that when T_c in a grid point location $\mathbf{x}_c$ is known, the gradient of T in $\mathbf{x}_c$ is given by

$$\nabla T_c = P^{-1}(\mathbf{r} T_c + \mathbf{s}).$$

Here $\mathbf{s}$ is a vector with elements that involve T-values on the upwind side of $\mathbf{x}_c$, while P and $\mathbf{r}$ only involve coordinates of grid points. The gradient of S_u and S_v in $\mathbf{x}_c$ can be expressed accordingly based on the corresponding grid point values of S_u and S_v on the upwind side of $\mathbf{x}_c$. We examine the two-dimensional case first, similarly to how ∇T was discretized in Sect. 3.1.3.

Let S denote the parameter distribution field in two dimensions, and let S_c be the unknown value of S in $\mathbf{x}_c$ (Fig. 4.3). Then

$$\nabla S_c = P^{-1}(\mathbf{r} S_c + \mathbf{s}') = P^{-1} \begin{pmatrix} r_a S_c + s'_a \\ r_b S_c + s'_b \end{pmatrix}, \tag{4.6}$$

where $\mathbf{s}' = (s'_a, s'_b)^T$ takes the same form as the vector $\mathbf{s}$ supporting the gradient ∇T_c, but with grid point values for S replacing those for T. The first and

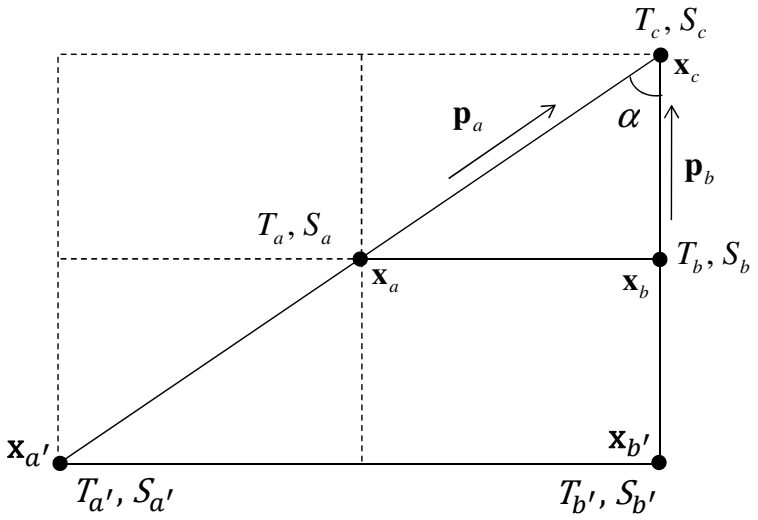


Fig. 4.3: The gradient ∇S_c in grid point position $\mathbf{x}_c$ is computed from known solution values S_a and S_b in $\mathbf{x}_a$ and $\mathbf{x}_b$, respectively, in the upwind direction from $\mathbf{x}_c$. The second order scheme also uses computed values $S_{a'}$ and $S_{b'}$.

second order difference operators for $\mathbf{s}'$ are (second order in parentheses)

$$s'_a = \frac{-S_a}{\|\mathbf{x}_c - \mathbf{x}_a\|} \quad \left(\frac{-4S_a + S_{a'}}{2\|\mathbf{x}_c - \mathbf{x}_a\|}\right),$$
$$s'_b = \frac{-S_b}{\|\mathbf{x}_c - \mathbf{x}_b\|} \quad \left(\frac{-4S_b + S_{b'}}{2\|\mathbf{x}_c - \mathbf{x}_b\|}\right).$$

When T_c is known, the field vector $\nabla_{\mathbf{p}} H_c$ through $\mathbf{x}_c$ can be expressed by the discretized version of Eq. (4.4) as

$$\nabla_{\mathbf{p}} H_c = F_{\mathtt{prop}} \frac{\nabla T_c}{\|\nabla T_c\|} + \psi_{\mathtt{adv}}\, \mathbf{a}.$$

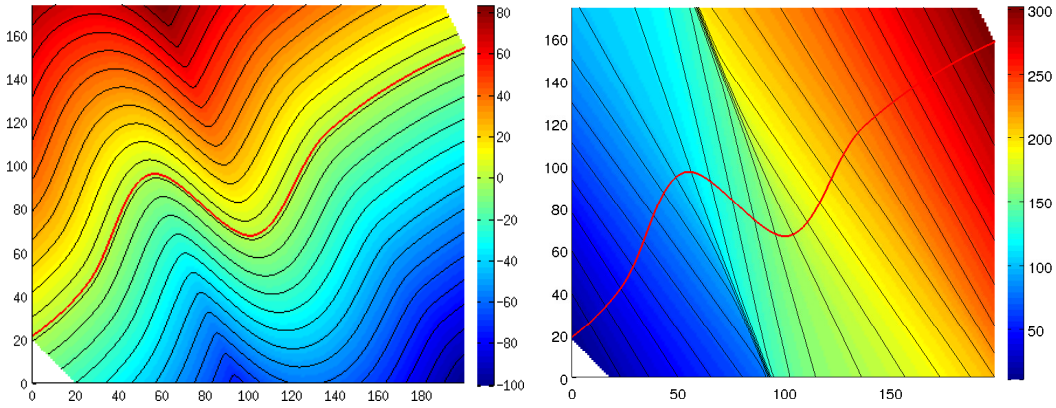
The orthogonality condition in Eq. (4.5) in $\mathbf{x}_c$ then becomes

$$P^{-1}(\mathbf{r}S_c + \mathbf{s}') \cdot \nabla_{\mathbf{p}} H_c = 0, \tag{4.7}$$

and solving for S_c we get

$$S_c = -\frac{\nabla_{\mathbf{p}} H_c \cdot P^{-1}\mathbf{s}'}{\nabla_{\mathbf{p}} H_c \cdot P^{-1}\mathbf{r}}. \tag{4.8}$$

The S-field is consistently computed on the same grid as the T-field by exactly the same upwind construction. Computation of a grid value S_c can take place



(a) (b)

Fig. 4.4: (a): T-field of a Class 1C fold modeled with $F_{\texttt{prop}} = 0.3$, $\psi_{\texttt{adv}} = 1$ and $\mathbf{a} = (-0.5, 1)$. (b): The corresponding S-field with isocurves. The parameterization of the curve runs from left to right.

immediately after T_c has been computed such that the correct upwind values T_a, T_b, S_a, S_b, and $T_{a'}, T_{b'}, S_{a'}, S_{b'}$ for second order differences, are available (Fig. 4.3). Conceptually, extension to three dimensions for computing $S_u(\mathbf{x}_c)$ and $S_v(\mathbf{x}_c)$ associated with a parametric surface $\Gamma_0(u, v)$ is straightforward, and can be explained following the same line of arguments as used for the three-dimensional T-field in Sect. 3.1.4. Fig. 4.4(b) shows an example of an S-field for a Class 1C fold in two dimensions. The isocurves of the S-field represent a field of dip isogons as a continuum in the whole domain.

4.2.1 Dependency Graph for Post-Computation of the S-field

From an application point of view, it might be necessary to compute the S-field (and other scalar fields that depend on the T-field) on demand in a post-process, and not along with the T-values in the front propagation algorithm. This can be consistently done if the stencils used for computing T-values are recorded. For example, the stencil in Fig. 4.3 for calculating T_c can be compactly stored as the quadruple $(-1, -1, 0, -1)$ for first order finite differences. The two first entries represent the offset in the grid from $\mathbf{x}_c$ to $\mathbf{x}_a$, and the two last entries represent the offset from $\mathbf{x}_c$ to $\mathbf{x}_b$. If second order finite differences are used, the quadruple is $(-2, -2, 0, -2)$ in this example. In three dimensions, the stencil is a tetrahedron with three stencil legs meeting at $\mathbf{x}_c$, and we need nine offset entries to store the stencil. To each grid point we can associate a dependency graph, which is a binary tree in two dimensions and a ternary tree in three dimensions. Note that S_c depends on grid point

values S_a and S_b (and $S_{a'}$ and $S_{b'}$ for second order differences) on the upwind side of $\mathbf{x}_c$. Therefore, post-computing grid point values in S must be done in proper order. This is unlike post-processing of, e.g., the gradient field ∇T by Eq. (3.6), which can be done in arbitrary order. One solution is to store the order in which T-values are computed and then compute S-values in the same order from known upwind values. Or, S-values can be computed recursively by starting from an arbitrary grid point $\mathbf{x}_c$ and following the dependency graph upwind until one reaches known values, or at the deepest level until one reaches the initial narrowband around Γ_0 where S-values are known from the initialization step. This recursive approach requires that the dependency graph is acyclic, but this is guaranteed since the stencils are constructed from grid points with known T-values.

4.3 Blending of Scalar Fields

Until now, T-fields and S-fields have been constructed from one single interface corresponding to the boundary condition in the front propagation problem given by Eq. (2.23). The propagating front, which corresponds to isosurfaces of T, has then been used to represent a continuum of layer boundaries. However, this continuum of layer boundaries will not, in general, align with other interfaces observed in the same fold.

Consider the sample fold in Fig. 4.5(a), with T-field T_1 generated from the interface Γ_1. Another observed interface Γ_2 in the same geological fold should then coincide with an isosurface of T_1, but this is not the case in the example since T_1 is computed independently of Γ_2. Let T_2 be the T-field generated from Γ_2, as shown in Fig. 4.5(b), with zero set at Γ_2 and negative values on that side which Γ_1 is located. Let $G_{T_1>0}$ and $G_{T_2<0}$ be the sets of grid points where T_1 is positive and T_2 is negative, respectively. We want to create a new field T_{12} on $G_{T_{12}} = G_{T_1>0} \cap G_{T_2<0}$ such that Γ_1 and Γ_2 coincide with two isosurfaces of T_{12}, and such that the isosurfaces of T_{12} have monotonically increasing t-values between the two interfaces as in Fig. 4.5(c). We first modify field T_2 with a constant value $t_{\texttt{lift2}} = \max |T_2 \in G_{T_{12}}|$ and set

$$\widetilde{T_2} = T_2 + t_{\texttt{lift2}}, \tag{4.9}$$

where the plus sign denotes the element-wise sum of all array elements with the scalar $t_{\texttt{lift2}}$. Then $\widetilde{T_2}$ is greater than zero everywhere in $G_{T_{12}}$, and $t_{\texttt{lift2}}$ is the value of the isosurface at Γ_2. T_{12} can be generated as a linear blend between T_1 and $\widetilde{T_2}$ using weights that resemble inverse distance weights. Let W_1 and W_2

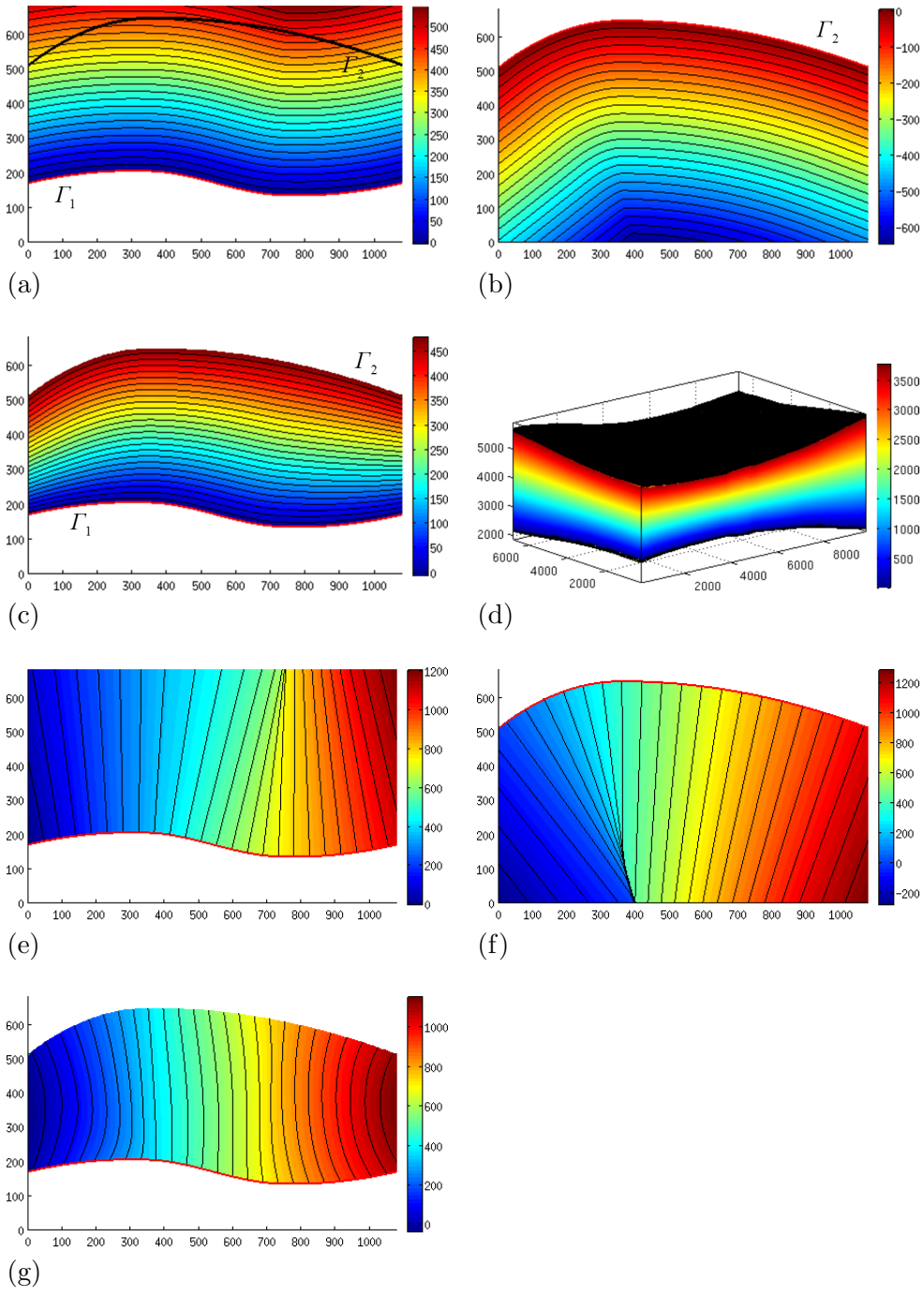


Fig. 4.5: Blending of scalar fields. (a): T-field generated from an interface Γ_1. Another interface Γ_2 is observed in the same fold. The two T-fields in (a) and (b), computed from interfaces Γ_1 and Γ_2, respectively, are blended to produce the combined T-field in (c). (d) shows an example of a blended T-field in 3D. (e) and (f) show the S-fields generated from Γ_1 and Γ_2, that are blended to produce the S-field in (g).

be scalar fields defined over the same grid as T_1 and T_2,

$$W_1 = -T_2 \,/\, (T_1 - T_2) \text{ and}$$
$$W_2 = -T_1 \,/\, (T_2 - T_1),$$

where the division sign denotes element-wise division. W_1 has zero-values at Γ_2 and monotonically increasing values away from Γ_2 towards Γ_1 where W_1 equals 1. Similarly, W_2 has zero-values at Γ_1 and monotonically increasing values towards Γ_2 where W_2 equals 1. The element-wise sums of the weights are 1 in the blending area. Then T_{12} can be computed by

$$T_{12} = T_1 \circ W_1 + \widetilde{T}_2 \circ W_2, \tag{4.10}$$

where $T_i \circ W_i$ denotes element-wise product of array elements. Fig. 4.5(c) shows the blended field T_{12}, and (d) shows an example of a blended T-field in 3D. When T_1 and T_2 are smooth, the weights are also smooth, and consequently the resulting field T_{12} is smooth. Other scalar fields derived from T_1 and T_2 can be blended on $G_{T_{12}}$ using the same weights. Fig. 4.5(e) and (f) show S-fields S_1 and S_2 generated from T_1 and T_2, respectively, and (g) shows the blended field

$$S_{12} = S_1 \circ W_1 + S_2 \circ W_2. \tag{4.11}$$

Note that this blending of T-fields and S-fields have geological relevance, like the computation of T_1 and T_2, which is based on information about the actual folding regime. Close to Γ_1 the blended field T_{12} takes on values that are approximately the values of T_1, and close to Γ_2 the values of T_{12} are approximately the values of T_2. Then the blending produces a smooth transition of the fields between Γ_1 and Γ_2. One should note that the example in Fig. 4.5 is exaggerated. In a real example the two interfaces Γ_1 and Γ_2 would be more similar in shape when they belong to the same fold and the corresponding geological time lag is not too big.

This blending scheme can be made more general to blend scalar fields assosiated with more than two interfaces. The weights can be constructed to resemble Lagrange polynomials such that a Lagrangiangian interpolation scheme can be written as

$$F = \sum_j F_j \circ W_j \tag{4.12}$$

to produce the blended field F from the fields F_j. For example, to blend three scalarfields we would need the three weights

$$W_1 = (T_2 \circ T_3) \,/\, [(T_1 - T_2) \circ (T_1 - T_3)]\,,$$
$$W_2 = (T_1 \circ T_3) \,/\, [(T_2 - T_1) \circ (T_2 - T_3)]\,, \text{ and}$$
$$W_3 = (T_1 \circ T_2) \,/\, [(T_3 - T_1) \circ (T_3 - T_2)]\,,$$

where T_1, T_2 and T_3 are T-fields from three non-overlapping interfaces. Blending of three T-fields and three S-fields can then be computed by

$$T_{123} = T_1 \circ W_1 + \widetilde{T}_2 \circ W_2 + \widetilde{T}_3 \circ W_3, \quad \text{and}$$
$$S_{123} = S_1 \circ W_1 + S_2 \circ W_2 + S_3 \circ W_3,$$

respectively, where $\widetilde{T}_2$ is given by Eq. (4.9), and $\widetilde{T}_3$ is constructed similarly such that it is positive everywhere in the blending area.

Blending of scalar fields are used in the restoration scheme presented in the next section. It can also be used to populate properties in space between two interfaces. Fig. 4.6 shows an example of distributing property data from

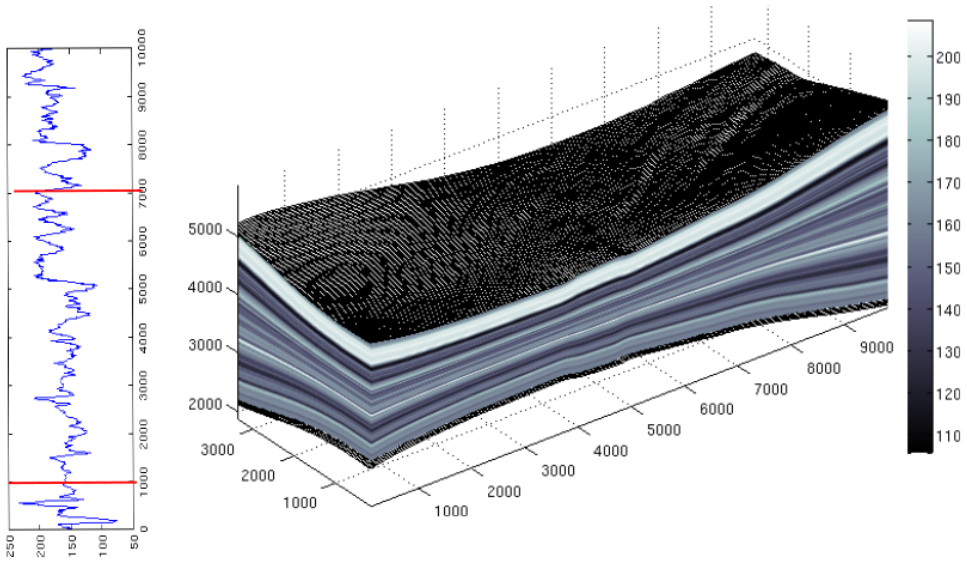


Fig. 4.6: Property function from a well log and property distribution in a volume between two layer boundaries.

a well log between two layer boundaries. The blended T-field of the volume to the right is shown in Fig. 4.5(d). Here we assume that there are no faults or unconformities present. First the two horizontal lines on the well log to the left are identified so that they correspond to the lower and upper interface of the 3D volume. The depth interval between the lines is mapped to $[t_1, t_2]$, where t_1 and t_2 are T-values corresponding to the lower and upper interface, respectively. Conceptually, property values can then be interpolated and represented by a univariate property function

$$f_\phi : t \longrightarrow \phi.$$

A grid point with value t_g from the blended T-field is then given the property value $\phi_g = f_\phi(t_g)$. This property distribution is without lateral variation in

the parameter space. Having more than one well log present may require correlation and blending of property data from the logs, and properties that are sampled from a trivariate function

$$f_\phi : (u, v, t) \longrightarrow \phi.$$

Thus, a grid point with parameter values (u_g, v_g, t_g), with $u_g \in S_u$ and $v_g \in S_v$, is given the property value $\phi_g = f_\phi(u_g, v_g, t_g)$.

4.4 Numerical Framework for Restoration and Reconstruction

Several methods have been proposed for restoration of geological volumes. To provide a point of reference for our method, which is based on the Hamilton-Jacobi framework and the parameterization of the 3D space presented in Sect. 4.1, a brief review of the principles underlying some alternative methods is first given.

4.4.1 Brief Overview of Some Existing Methods

Maerten and Maerten (2006) and Moretti *et al.* (2006) use principles of continuum mechanics and finite element formulations to compute the restored model. First, a tetrahedral mesh is constructed to represent the structural model. This mesh must conform to the geological interfaces such that no edge of the mesh intersects an interface. An example in 2D is shown in Fig. 4.7(a). Geomechanical properties for rock units and appropriate displacement boundary conditions are then associated to mesh nodes. For example, the displacement of a rock block along a fault surface is given by specifying source and destination locations of some of the mesh nodes, and a selection of other nodes may be constrained to have a fixed location. The same type of constraints are used for unfolding horizons to recover an original depositional system. The restored structural model with strain and stress fields are then computed by solving the equation system resulting from the finite element formulation. The system in Maerten and Maerten (2006) is solved by the Gauss-Seidel method, while the system in Moretti *et al.* (2006) is solved by a Newton-Rapson iteration scheme (Golub and Loan (1996)). Recently, Durand-Riard *et al.* (2010) proposed an approach where faults and horizons are implicitly defined as distances to nodes of the tetrahedral mesh. Then the mesh need not be constrained to conform to fault surfaces and horizons. Another approach based on geomechanical principles was presented in Santi and Martha (2003) for restoration of faults in geological

cross-sections. This method employs dynamic relaxation (Underwood, 1983) coupled with the finite element method as a numerical scheme to compute the restored model. The methods above rely on linear elastic material behavior, but heterogeneous properties can be assigned to honor vertical and lateral variations in rock rheology.

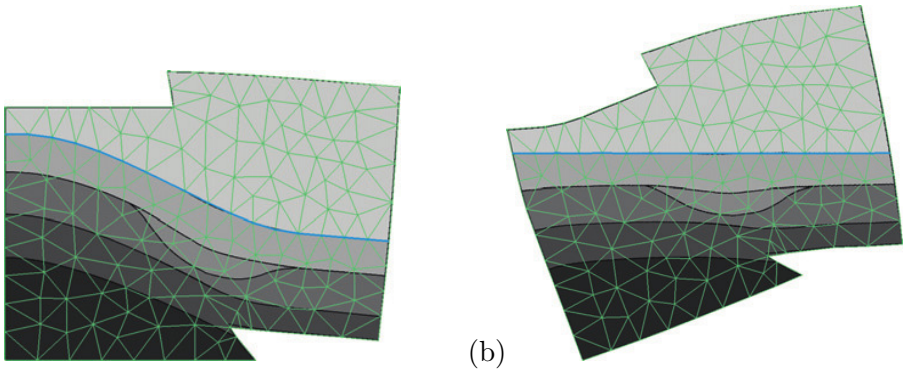


Fig. 4.7: (a): Unrestored structural model with a triangular mesh conforming to a reference horizon to be unfolded. (b): Restored model with the unfolded horizon and the transformed mesh. (Illustration from Gilardet *et al.* (2013); courtesy of Mathieu Gilardet.)

Geomechanical properties are not always available, in particular at an early stage of the interpretation workflow. A simple approach to seismic image restoration of cross-sections, without the need to define rock properties, is given in Gilardet *et al.* (2013). This approach is based directly on a general purpose geometric shape deformation framework borrowed from computer graphics (Weng *et al.*, 2006). Similar to the methods by Maerten and Maerten and Moretti *et al.*, this method requires a mesh (2D triangulation) conforming to geological interfaces. De-faulting and unfolding horizons are controlled by pure geometric constraints, and deformations are defined as pure geometric transformations of the mesh without taking into account the physical properties of the rock. Instead of taking into account rheological behavior, the algorithm aims to preserve two local shape properties of the mesh: the relative position of nodes on the boundary of the mesh expressed by Laplacian coordinates, and the relative position of interior nodes expressed by mean value coordinates (Floater, 2003). The set-up results in a non-linear least squares optimization problem that is solved by a Gauss-Newton iteration scheme. Fig. 4.7 shows an example of unfolding a horizon using this method. The transformation is intrinsically bijective such that the inverse transformation can be defined in a reconstruction process.

4.4.2 Data Restoration and Model Reconstruction Based on the Hamilton-Jacobi Framework

The parameterization derived in Sect. 4.1 and 4.2 provides us with a powerful tool for restoration of geological volumes both in 2D and 3D. Contrary to the finite element based methods mentioned in the previous section, the method we derive here does not require geomechanical properties of the rock as input. Nevertheless, the method makes physical and geological sense since the underlying parameterization controlling deformations reflects the actual folding regime of the geological volume. Another advantage is that the method is meshless, thus eliminating the need for constructing a complex tetrahedral mesh to represent the geological model. All calculations of deformations take place on the regular finite difference grid by utilizing the T-field, and the S_u- and S_v-fields computed by the numerical scheme in Sect. 4.2. This yields fast and stable numerical computations and simple concepts for the user for setting up the restoration workflow.

To explain the main ideas we start with simple unfolding of a horizon, and for ease of explanation we look at the two-dimensional case first. Let $\Gamma_0(u)$ and $\bar{\Gamma}_0(u)$ be parametric curves corresponding to a horizon before and after restoration, respectively, as shown in Fig. 4.8. Geometrical constraints are imposed such that a selection of source points $\{\mathbf{x}(u_p, t = 0)\}$ on $\Gamma_0(u)$ corresponds to specified target locations $\{\bar{\mathbf{x}}(u_p, t = 0)\}$ on the restored horizon $\bar{\Gamma}_0(u)$. That is, a particle of sediment at location $\mathbf{x}(u_p, 0)$ will be repositioned to location $\bar{\mathbf{x}}(u_p, 0)$ in Euclidean space during restoration. For example, the set $\{\mathbf{x}(u_p, 0)\}$ may be the start and end points of $\Gamma_0(u)$, and optionally some points in the interior of the curve. Due to extensional or compressional forces that may have acted over geological time, the curves may have different lengths, but the curves are properly parameterized such that the parameter values $\{u_p\}$ corresponding to points on Γ_0 and $\bar{\Gamma}_0$ specified by the user are preserved. Thus, we have a mapping of a set of distinct source points to target points on the two curves

$$\{\mathbf{x}(u_p, 0)\} = \{\Gamma_0(u_p)\} \longrightarrow \{\bar{\Gamma}_0(u_p)\} = \{\bar{\mathbf{x}}(u_p, 0)\}.$$

Similarly, we assume that any point on $\Gamma_0(u)$ can be associated with a point on $\bar{\Gamma}_0(u)$ by the mapping

$$\mathbf{x}(u, 0) = \Gamma_0(u) \longrightarrow \bar{\Gamma}_0(u) = \bar{\mathbf{x}}(u, 0).$$

Mapping of a particle with location (u, t), where $t < 0$ (i.e., the particle is positioned away from Γ_0 and is older than particles on Γ_0) follows the same principle: the particle is repositioned to a new position $\bar{\mathbf{x}}(u, t)$ in Euclidean

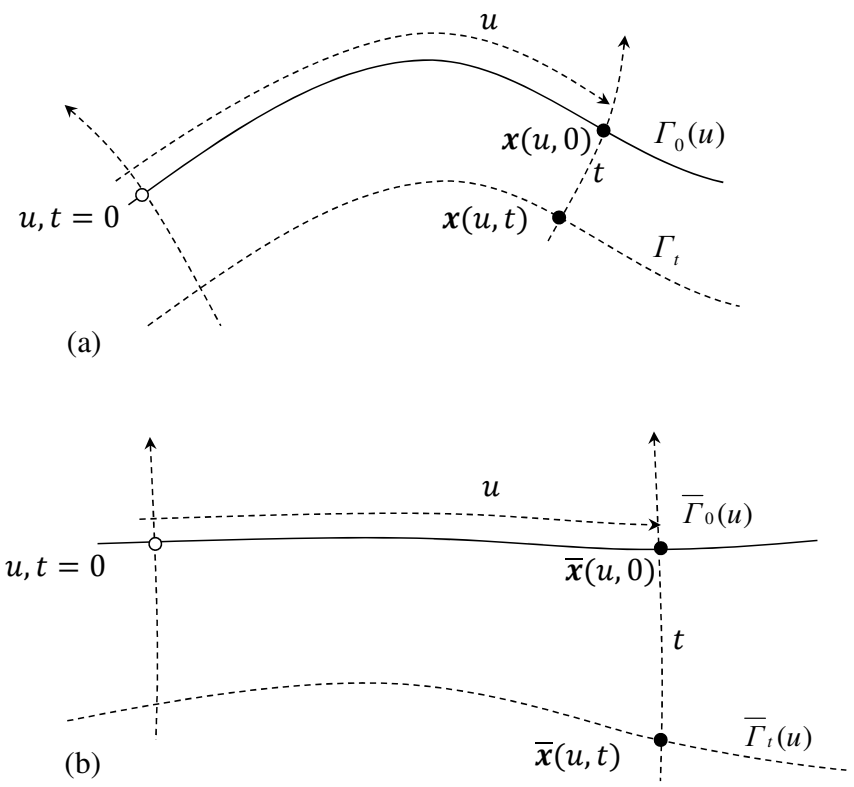


Fig. 4.8: Restoration of a layer between two horizons $\Gamma_0(u)$ and $\Gamma_t(u)$. (a): Before restoration with a particle of sediment at location $\mathbf{x}(u,0)$ on the horizon, and another older particle located at $\mathbf{x}(u,t)$. (b): After restoration where the particles have been moved to new positions in the Euclidean space, but the location (u,t) in the parameter space of each particle is preserved.

space, but the particle's location (u,t) in the parameter space is always preserved by the mapping

$$\mathbf{x}(u,t) \longrightarrow \bar{\mathbf{x}}(u,t).$$

The model space is typically populated with properties such as porosity, permeability and oil saturation, or velocity and density needed for calculation of seismic responses and generation of synthetic seismics. Properties can also be seismic amplitudes in the case of seismic image restoration. In a restoration process following the principles outlined above, one or more property values are associated to $\mathbf{x}(u,t)$ and thus mapped along with the particle of sediment to a new position $\bar{\mathbf{x}}(u,t)$ in the Euclidean space. Properties can of course change over geological time by post-depositional transformations such as diagenesis and metamorphism, but we assume here that a property value at $\bar{\mathbf{x}}(u,t)$ can

be derived from the corresponding property at $\mathbf{x}(u,t)$ before restoration. Let G and $\bar{G}$ denote the $m \times n$ and $\bar{m} \times \bar{n}$ regular grids of the unrestored and restored volume, respectively. Recall that the parameterization expressed by the S-field and the T-field of the geological volume is distributed on the regular grids G and $\bar{G}$, say, as (S,T) and $(\bar{S},\bar{T})$. Similarly, a set of property values, say Φ, is distributed as a scalar field on the unrestored volume. Thus, on the grid G we have the three data sets

$$(S,T,\Phi) = \{(u,t,\phi)_{i,j}\},\ i \in \mathcal{I}_{\mathbf{x}},\ j \in \mathcal{I}_{\mathbf{y}}, \tag{4.13}$$

where $\mathcal{I}_{\mathbf{x}}$ and $\mathcal{I}_{\mathbf{y}}$ are grid indices given in Eq. (4.3). These scalar fields exist in regular grid positions in the Euclidean space, but in the parameter space the property field Φ exists in scattered locations $\{(u,t)_{i,j}\}$. In the restoration process we want to map the property values to the restored volume and populate $\bar{G}$ with new properties. Thus, we want to establish

$$(\bar{S},\bar{T},\bar{\Phi}) = \{(\bar{u},\bar{t},\bar{\phi})_{i,j}\},\ i \in \bar{\mathcal{I}}_{\mathbf{x}},\ j \in \bar{\mathcal{I}}_{\mathbf{y}},$$

where $\bar{S}$ and $\bar{T}$ are known. Since a grid point location in G does not map to a grid point location in $\bar{G}$ in general, we must resample properties in locations $\{(\bar{u},\bar{t})_{i,j}\}$ of the regular grid $\bar{G}$ to find the mapped properties. Based on S, T and the existing property field Φ on G, given by Eq. (4.13), we define a bivariate property function

$$f_\phi : (u,t) \longrightarrow \phi,$$

that can be evaluated in an arbitrary parameter pair (u,t). The construction of f_ϕ is a scattered data interpolation problem. Given distinct locations $\{(u,t)_{i,j}\}$ in the parameter space and corresponding data values $\{\phi_{i,j}\}$, we seek the surface $f_\phi(u,v)$ (hypersurface in 3D) that approximates the data values

$$f_\phi(u,t)_{i,j} \approx \phi_{i,j}, \quad i = 1,\ldots,m \quad j = 1,\ldots,n.$$

Since locations of particles in the parameter space are preserved, the mapped property field $\bar{\Phi}$ can be found by evaluating f_ϕ in all grid points of $\bar{G}$, thus

$$\bar{\Phi} = \{\bar{\phi}_{i,j}\} = \{f_\phi(\bar{u},\bar{t})_{i,j}\},\ i \in \bar{\mathcal{I}}_{\mathbf{x}},\ j \in \bar{\mathcal{I}}_{\mathbf{y}}.$$

The construction of f_ϕ can be computational demanding when m and n are large. We used multilevel B-splines (Lee *et al.*, 1997; Hjelle, 2001) for generating f_ϕ, which is fast and stable also in the trivariate case when G and $\bar{G}$ are 3D grids. The multilevel method in Appendix A produces a smoother function f_ϕ when the data are irregular spaced, but there is no implementation of this method in 3D. Both methods produce a tensor product function f_ϕ that is fast to evaluate for finding the mapped properties on $\bar{G}$.

To illustrate how deformation of material takes place in a 2D section by the mapping described above, we use a simple property field Φ consisting of a pattern of black circles on a white background. Fig. 4.9(a) illustrates a flat layer with no initial compaction or strain around a reference curve Γ_1. The T- and S-fields are generated by $F_{\texttt{prop}} = 1.0$ and $\psi_{\texttt{adv}} = 0$. ($F_{\texttt{prop}} = 0$ and $\psi_{\texttt{adv}} = 1.0$ would produce the same result since Γ_1 is horizontal.) The fold in (b) simulates a parallel Class 1B fold with T- and S-fields generated by $F_{\texttt{prop}} = 1.0$ and $\psi_{\texttt{adv}} = 0$ based on a reference curve Γ_2. In (c) and (d) the u- and t-lines of the parameterization of the folds in (a) and (b) are shown. The property pattern Φ in (a) is mapped to $\bar{\Phi}$ in (b) in agreement with the scheme above. The resulting strain field for the fold in (b) can be characterized by compression parallel to layer boundaries below the anticline, and stretching above the anticline (away from from the hinge). For the syncline, the opposite applies. Γ_2 is a neutral curve with no tangential strain. Moreover, straight lines orthogonal to the neutral line remain orthogonal and preserve arc lengths during fold formation with no tangential strain along the lines (Fig. 4.9(d)). This folding regime corresponds to orthogonal flexure folding, also called neutral surface folding (Twiss and Moores, 2007). The strain pattern is also said to be tangential longitudinal (Ramsay and Huber, 1987; Bobillo-Ares *et al.*, 2006). The pattern in (e) illustrates pure compaction (very simplified) of the layer in (a) with the T- and S-fields in (e) generated by $F_{\texttt{prop}} = 0.6$ and $\psi_{\texttt{adv}} = 0$ from the same curve as in (a).

Note that only T-fields, S-fields and the original property field Φ are involved in the mapping above, while the geometries of the interfaces are only used to compute T-fields and S-fields. In many practical cases one is interested in mapping a layer delimited by two existing interfaces Γ_1 and Γ_2 to a new restored layer delimited by two interfaces $\bar{\Gamma}_1$ and $\bar{\Gamma}_2$ specified by the user. Then the fields S, $\bar{S}$ and T, $\bar{T}$ must be the blended scalar fields computed by Eq. (4.10) and Eq. (4.11), respectively. This can be generalized to map between layers delimited by more than two interfaces by producing blended T and S-fields as described by the a Lagrangiangian interpolation scheme in Eq. (4.12).

Conceptually, extension to three dimensions is straightforward. The interface between two layers is then a parametric surface $\Gamma(u, v)$ leading to two S-fields S_u and S_v on 3D grids G and $\bar{G}$. In particular, on the $m \times n \times p$ grid G we have the four scalar fields

$$(S_u, S_v, T, \Phi) = \{(u, v, t, \phi)_{i,j,k}\},\ i \in \mathcal{I}_{\mathbf{x}},\ j \in \mathcal{I}_{\mathbf{y}},\ k \in \mathcal{I}_{\mathbf{z}},$$

and on $\bar{G}$ we want to establish

$$(\bar{S}_u, \bar{S}_v, \bar{T}, \bar{\Phi}) = \{(\bar{u}, \bar{v}, \bar{t}, \bar{\phi})_{i,j,k}\},\ i \in \bar{\mathcal{I}}_{\mathbf{x}},\ j \in \bar{\mathcal{I}}_{\mathbf{y}},\ k \in \bar{\mathcal{I}}_{\mathbf{z}},$$

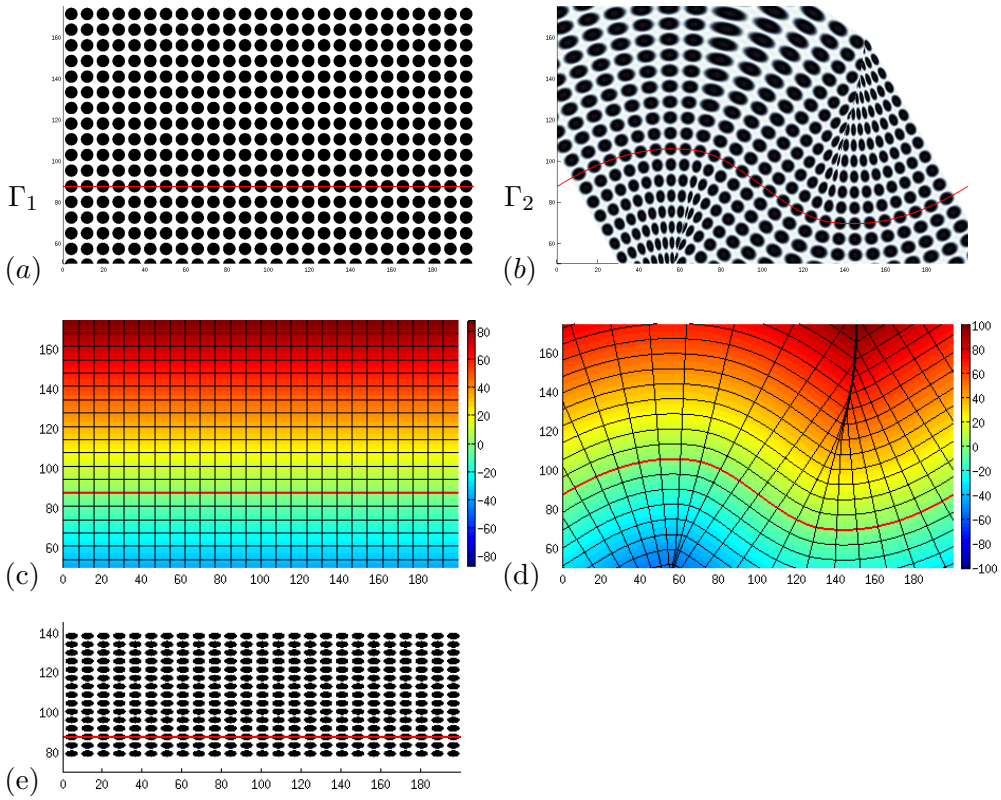


Fig. 4.9: Strain pattern from applying the proposed mapping from a horizontal layer to a folded layer. (a): Horizontal layer with no initial compaction or strain. (b): The strain pattern after orthogonal flexure folding to a Class 1B fold. (c): Parameterization, u-lines (horizontal) and t-lines (vertical) overlaid the T-field of the horizontal fold. (d): Parameterization and T-field of the folded layer. (e): Illustration of strain pattern after compaction of the horizontal layer.

where $\bar{S}_u$, $\bar{S}_v$ and $\bar{T}$ are known. The property function is then a trivariate function

$$f_\phi(u, v, t) \longrightarrow \phi \tag{4.14}$$

constructed by a trivariate version of the multilevel B-spline algorithm (Lee *et al.*, 1997). The mapped property field $\bar{\Phi}$ is found by evaluating f_ϕ in all grid points of $\bar{G}$, thus

$$\bar{\Phi} = \{\bar{\phi}_{i,j,k}\} = \{f_\phi(\bar{u}, \bar{v}, \bar{t})_{i,j,k}\},\ i \in \bar{\mathcal{I}}_{\mathbf{x}},\ j \in \bar{\mathcal{I}}_{\mathbf{y}},\ k \in \bar{\mathcal{I}}_{\mathbf{z}}. \tag{4.15}$$

Fig. 4.10 shows an example of 3D seismic image restoration implemented in Statoil ASA's Compound Earth Simulator (CES) using the scheme presented above. The reverse mapping, that is, the construction process forward in geological time, can be described symmetrically by exchanging G and $\bar{G}$. The different steps in restoration of a single horizon can be summarized as follows (consult Fig. 4.8 and 4.10).

(i) Identify horizon Γ_0 in the seismic image and generate a set $\mathcal{P}$ of points (e.g. by autotracking) laying on Γ_0.

(ii) Fit a surface $\Gamma_0(u, v)$ to $\mathcal{P}$, e.g. by using multilevel least squares approximation (Appendix A).

(iii) Define the target horizon and construct the surface $\bar{\Gamma}_0(u, v)$. This may require specialized interactive software modeling tools, or the surface may exist in the workflow of another geomodeling system.

(iv) Compute scalar fields (S_u, S_v, T) from $\Gamma_0(u, v)$, and $(\bar{S}_u, \bar{S}_v, \bar{T})$ from $\bar{\Gamma}_0(u, v)$.

(v) Establish the trivariate property function Eq. (4.14) from (S_u, S_v, T) and the property field Φ of the unrestored volume, e.g. by using multilevel B-splines (Lee *et al.*, 1997).

(vi) Find the mapped property field $\bar{\Phi}$ of the restored volume by the sampling in Eq. (4.15).

Successive application of restoration or reconstruction of horizons in a geological volume as proposed here, combined with restoration of faults (Sect. 4.4.4), realizes the theoretical concept of Earth Recursion (Petersen and Hjelle, 2008) in the dR/mR-workflows in CES. Fig. 4.14 on page 87 shows folding of a volume after a sequence of fault processes. More examples of data restoration and model reconstruction on realistic volumes can be found in Petersen *et al.* (2012).

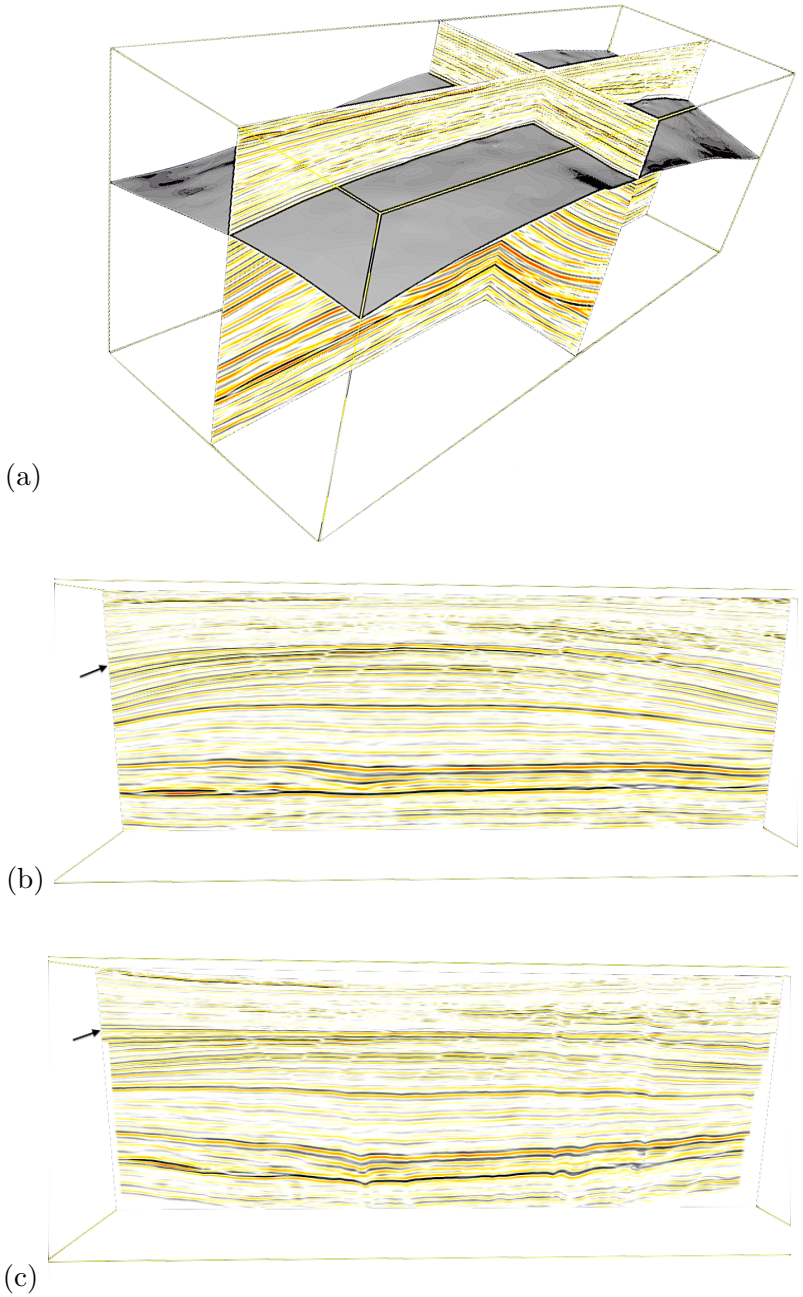


Fig. 4.10: Restoration of a 3D seismic volume: (a): A horizon and two seismic cross sections originating from offshore Angola. (b): The leftmost–rightmost cross section from (a) before restoration. (c): The same cross section after restoration. Arrows on the left indicate the horizon that is unfolded. The horizion in (a) was modeled from autotrack data with the multilevel method in Appendix A. (Screen shots from Statoil ASA's Compound Earth Simulator (CES).)

4.4.3 Discussion

The Hamilton-Jacobi mathematical framework for modeling the different fold classes as a propagating front, presented in Chapter 2, was primarily designed to reflect the geometry of the present day fold, and not to reflect physical processes of fold formation in general. Consequently, restoration by the mapping presented above does not always reflect the actual strain pattern of a natural fold, and particles are not necessarily mapped to correct positions in the Euclidean space. Nevertheless, as demonstrated above by the Class 1B fold formed under orthogonal flexure folding, the strain pattern agrees reasonably well with that of a natural fold when the parameterization is constructed directly from the Hamilton-Jacobi framework. The reason is the choice of the t-lines of the parameterization that were aligned with the group velocity vector $\nabla_{\mathbf{p}} H$ in Eq. (4.4), and that $\nabla_{\mathbf{p}} H$ points in the direction of the dip isogons. In this particular example with folding of a competent layer, the dip isogons deform indeed as material lines (Srivastava and Shah, 2008). Since the locations of particles in the parameter space are preserved under the mapping, the strain pattern agrees well with that of natural folding.

Consider a standardized upright antiform of Class 1B type formed under orthogonal flexure folding, as for the left half of the fold in Fig. 4.9(b) and (d). Suppose that homogeneous strain is superimposed in the upward axial direction. Then thinning of the limbs, thickening of the hinge, and less convergent dip isogons result (Srivastava and Shah, 2008). The result is a Class 1C fold (Fig 2.1). Further modification by increasing the homogeneous strain, will approach it towards a Class 2 fold (but remain a Class 1C fold). Folds formed in this way are also called flattened parallel folds. In our front propagation model we obtain the Class 1C fold type by $F_{\texttt{prop}} > 0$ and $\psi_{\texttt{adv}} > 0$, and by decreasing the $F_{\texttt{prop}}/\psi_{\texttt{adv}}$ ratio the fold approaches Class 2 type for which $F_{\texttt{prop}} = 0$ (Table 2.2). If we assume that dip isogons deform as material lines under such flattening, we may expect the modeled strain pattern to agree well with that of a natural fold.

On the other hand, suppose that homogeneous strain is superimposed outward and orthogonal to the axial direction of the standardized Class 1B fold formed under orthogonal flexure folding. Then thinning of the hinge, thickening of the limbs, and more convergent dip isogons result. The result of the flattening is a Class 1A fold. In our front propagation model we obtain the Class 1A fold type by $F_{\texttt{prop}} > 0$ and $\psi_{\texttt{adv}} < 0$. If we assume that dip isogons still deform as material lines, we may expect that the modeled strain pattern agrees well with that of a natural fold also in this case.

These observations justify the choice of aligning t-lines of the parameteri-

zation with $\nabla_{\mathbf{p}} H$ (Eq. (4.4)), at least for the deformation types demonstrated above. And, by the principle of preserving the locations of particles in the parameter space under the mapping derived in Sect. 4.4.2, particles and properties are mapped to correct positions in the Euclidean space in the restoration process. Other folding mechanisms, many of which can be very complex, may require other directions of t-lines, but using the direction of $\nabla_{\mathbf{p}} H$ appears reasonable if $F_{\texttt{prop}}$, $\psi_{\texttt{adv}}$ and $\mathbf{a}$ are chosen carefully. In addition, the blending scheme in Sect. 4.3 can be used to generate more complex S and T fields for a layer delimited by two interfaces. In a flexural slip fold, material lines will be discontinuous over bedding planes on the fold limb. Then each layer between two bedding planes must be modeled individually if we want the model to reflect the actual strain pattern, possibly by blending S-fields and T-fields as described above.

4.4.4 Modeling and Restoration of Faults

Faulting involves displacement of material in a volume of rock. This may happen simultaneously with folding and other geological processes, but in the following we regard faulting as a separate process in a sequence of geological processes. We assume that displacement of material takes place inside a portion of the volume denoted the influence region (Fig. 4.11). Our meshless volumetric approach suggests that a fault be modeled as a 3D vector displacement field inside the influence region. Thus, a vector representing the relative displacement of material in space will be associated to each grid point inside the influence region. For restoration and reconstruction of faults we utilize the mathematical and numerical frameworks derived above for restoration and reconstruction of horizons. As it turns out, after setting up the parameterization of the 3D space relative to the fault surface with proper scalar fields (S_u, S_v, T) and $(\bar{S}_u, \bar{S}_v, \bar{T})$ for the unrestored and the restored volume, respectively, mapping of a property field Φ to $\bar{\Phi}$ when restoring a fault is performed exactly as summarized in the end of Sect. 4.4.2 for restoration of a horizon. In the following, we first derive the vector displacement field for representing a fault. For ease of explanation we first make some assumptions and simplifications, but in the end of the section we indicate how to generalize this in different ways.

Consider a dip-slip listric fault[1] where the fault surface is represented as a parametric surface $\Gamma_0(u, v)$. Assume that the maximum displacement vector $\vec{\delta}_{\max}$ is located at a point $\mathbf{x}_e$ on the fault surface and that displacements

[1] A listric fault is a fault with a curved fault plane, concave upwards. A dip-slip fault has its primary movement in the dip direction of the fault surface.

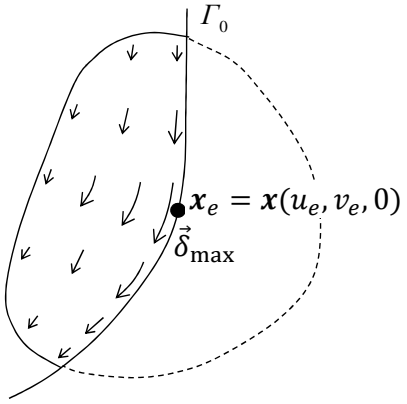


Fig. 4.11: Displacement vectors inside the influence region of a dip-slip listric fault in a 2D section.

gradually decrease away from $\mathbf{x}_e$ reaching zero at the boundary of the influence region (Fig. 4.11). Without loss of generality, assume that $\Gamma_0(u, v)$ is parameterized such that the u-line through $\mathbf{x}_e$ is aligned with the slip direction of the fault. The T-field is constructed relative to Γ_0 such that Γ_0 corresponds to the zero set of T. A natural choice is to let T be an isotropic distance field with $F_{\texttt{prop}} = 1.0$ and $\psi_{\texttt{adv}} = 0$ in Eq. (2.23). If Γ_0 has high curvature, a smaller value for $F_{\texttt{prop}}$ and a positive value for $\psi_{\texttt{adv}}$, and axial direction $\mathbf{a}$ pointing in the normal direction of the surface at $\mathbf{x}_e$, may be a better choice to guarantee that t-lines do not intersect inside the influence region. We return to this below. Since the T-field is zero at Γ_0, the scalar fields S_u and S_v yield u-lines and v-lines aligned with Γ_0, and with proper choice of parameters when constructing T, the curvilinear (u, v, t)-grid is well defined with non-intersecting and separated parameter lines inside the influence region as in the 2D example in Fig 4.12. In particular, if $\psi_{\texttt{adv}} = 0$ it follows from Eq. (4.4) and (4.5) that t-lines are orthogonal to both u-lines and v-lines. In the vicinity of $\mathbf{x}_e$ the vector displacement field is parallel to the fault surface and directed in the slip direction, and $\vec{\delta}_{\max}$ is aligned with the u-line through $\mathbf{x}_e$. If the deformation inside the influence region is of simple shear type, we may assume that the whole vector displacement field is aligned with the u-lines of the parameterization.

We define the shape of the influence region as an ellipsoid in the parameter space with center in

$$\mathbf{p}_e = (u_e, v_e, 0),$$

corresponding to $\mathbf{x}_e = \mathbf{x}(u_e, v_e, 0)$ in the Euclidean space, with its semi-major axis aligned with the u-line in the slip direction, and semi-minor axes aligned with the v-line and the t-line through $\mathbf{p}_e$. So, when drawn in the Euclidean space the ellipsoid is deformed in general, as shown in Fig. 4.12 of a listric

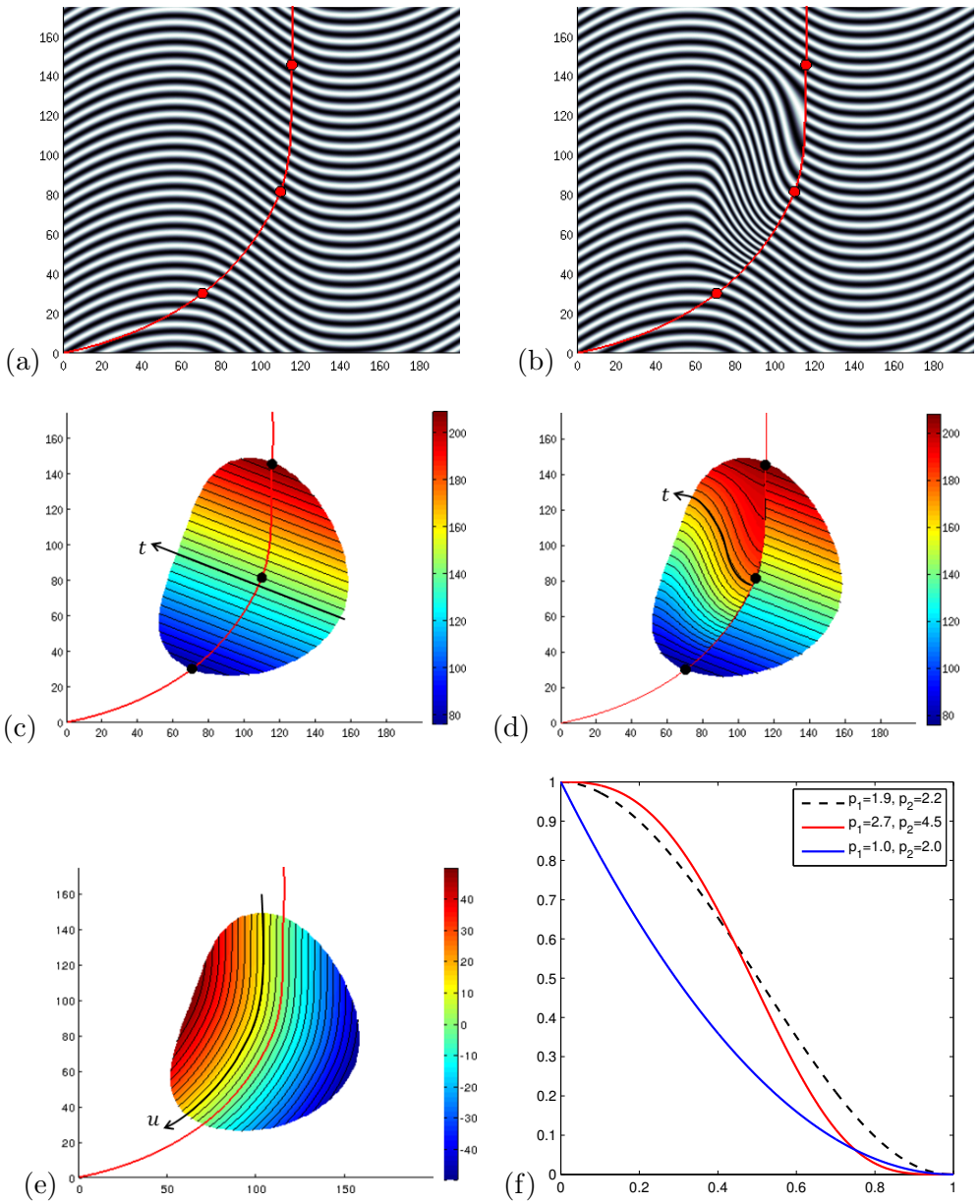


Fig. 4.12: (a): A sample property field and a fault trace in a 2D section of a folded structure before faulting. (b): Faulting downwards along one side of the fault trace based on simple shear. Extension of material above the epicenter of the fault, and compression below. (c) and (d): S_u-fields inside the influence region before and after faulting, respectively. The isolines of S_u are t-lines of the parameterization. (e): The T-field with isolines, which represent u-lines of the parameterization. (f): Examples of the cumulative distribution function in Eq. (4.16), with different parameters p_1 and p_2 controlling the shape.

fault. The lengths of the semi-axes must be set so that the ellipsoid in the parameter space approximates the observed influence region in the Euclidean space.

To distribute displacements inside the influence area, we define bell shaped functions with value = 1.0 in $\mathbf{p}_e$, and with decaying values away from $\mathbf{p}_e$ vanishing at the periphery. This bell shape can be resembled by the cumulative distribution function

$$\tilde{d}_{\texttt{bell}}(r) = \begin{cases} (1 - r^{p_1})^{p_2}, & 0 \leq r < 1 \\ 0, & r \geq 1, \end{cases} \tag{4.16}$$

where r is the distance in the parameter space from $\mathbf{p}_e$, normalized such that $r = 1.0$ on the periphery of the ellipsoid, and p_1 and p_2 are parameters for controlling the shape. If $p_1 > 1$ and $p_2 > 1$, the derivative is zero in $\mathbf{p}_e$ and at the periphery. In particular, if $p_1 \approx 1.9$ and $p_2 \approx 2.2$, the function approximates a sinusoid. If $p_1 = 1.0$ and $p_2 = 1.0$ we get the linear function $1 - r$. When p_1 increases from 1.0, a wider flattening occurs from $r = 0$, and when p_2 increases from 1.0, a wider flattening occurs at $r = 1.0$. Some examples of $\tilde{d}_{\texttt{bell}}(r)$ are shown in Fig. 4.12(f). We want the distribution function to depend on direction in space. This can be achieved by specifying three functions of the form defined in Eq. (4.16), one for each of the u, v and t-directions, with different parameters p_1 and p_2, say

$$d^u_{\texttt{bell}}(r_u), \quad d^v_{\texttt{bell}}(r_v), \quad d^t_{\texttt{bell}}(r_t).$$

Here, r_u, r_v and r_t take the role of r in Eq. (4.16) in each of the parameter directions. These functions can then be combined by a (tensor) product to produce the distribution function

$$d_{\texttt{bell}}(u, v, t) = d^u_{\texttt{bell}}(r_u) \cdot d^v_{\texttt{bell}}(r_v) \cdot d^t_{\texttt{bell}}(r_t).$$

Alternatively, displacements in the fault surface can be modeled as

$$d_{\texttt{bell}}(u, v, 0) = d^u_{\texttt{bell}}(r_u) \cos^2 \alpha + d^v_{\texttt{bell}}(r_v) \sin^2 \alpha,$$

where α is the angle relative to the u-axis, and the combined function as

$$d_{\texttt{bell}}(u, v, t) = \left(d^u_{\texttt{bell}}(r_u) \cos^2 \alpha + d^v_{\texttt{bell}}(r_v) \sin^2 \alpha\right) d^t_{\texttt{bell}}(r_t).$$

Both alternative constructions of $d_{\texttt{bell}}(u, v, t)$ open up for modeling complex and geologically relavant displacement patterns. The latter is similar to an approach published recently in Laurent *et al.* (2013). The scalar displacement field inside the influence region is then

$$d(u, v, t) = |\vec{\delta}_{\max}| \cdot d_{\texttt{bell}}(u, v, t), \tag{4.17}$$

where u, v and t are known in the grid points of the regular grid embedding the fault from S_u, S_v and T, respectively. We discretize the distribution function on the grid by

$$D_{\texttt{bell}} = \{d_{\texttt{bell}}(u,v,t)_{i,j,k}\},\ i \in \mathcal{I}_{\mathbf{x}},\ j \in \mathcal{I}_{\mathbf{y}},\ k \in \mathcal{I}_{\mathbf{z}}.$$

As a consequence of the orthogonality properties P1 and P2 in Sect. 4.1, the directions of the u-lines are $\nabla S_v \times \nabla T$. Then the vector displacement field can be written

$$\vec{D} = |\vec{\delta}_{\max}| \cdot D_{\texttt{bell}} \circ \frac{\nabla S_v \times \nabla T}{\|\nabla S_v \times \nabla T\|}. \tag{4.18}$$

Here, $\circ$ stands for the element-wise product of the 3D array on the left hand side with each of the three components of the gradient vector field on the right hand side. If T is an isotropic distance field, the u-lines are aligned with the gradient field of S_u, and Eq. (4.18) can be simplified accordingly by ∇S_u replacing $\nabla S_v \times \nabla T$.

So, how can we impose a fault on a property field of an un-faulted model, or conversely, how can we restore a fault by removing the impact of the fault on recorded seismic data? To restore a fault we need to reverse the vector displacement field represented by Eq. (4.18), that is, displacements $-\vec{D}$ must be applied to grid points inside the influence region. As for restoration of a horizon in Sect. 4.4.2, we have the scalar fields

$$(S_u, S_v, T, \Phi) = \{(u,v,t,\phi)_{i,j,k}\},\ i \in \mathcal{I}_{\mathbf{x}},\ j \in \mathcal{I}_{\mathbf{y}},\ k \in \mathcal{I}_{\mathbf{z}},$$

where Φ is the property field, for example recorded seismic data. We want to find the restored property field $\bar{\Phi}$, which will differ from Φ in grid points inside the influence region of the fault. Since we assume that the vector displacement field is of simple shear type, a property value $\bar{\Phi}(u,v,t)$ should take its value from $\Phi(u+\delta_u,\ v, t)$, where δ_u is an increment in the u-direction in the parameter space that corresponds to the displacement value from Eq. (4.17) in the Euclidean space. Suppose that the parameterization of the fault surface $\Gamma_0(u,v)$ is scaled such

$$\delta_u(u_e, v_e, 0) \approx |\vec{\delta}_{\max}|. \tag{4.19}$$

Then we can prepare the mapping of properties from Φ to $\bar{\Phi}$ by perturbing S_u and preserving S_v and T, thus

$$\begin{aligned} \bar{S}_u &= S_u - \delta_u(u_e, v_e, 0) \cdot D_{\texttt{bell}}, \\ \bar{S}_v &= S_v, \quad \text{and} \\ \bar{T} &= T. \end{aligned}$$

By the property in Eq. (4.19), we can also assume that the perturbations $\bar{S}_u - S_u$ are approximately Euclidean lengths. Fig. 4.12(c) and (d) show S_u and $\bar{S}_u$,

and how the t-lines of the parameterization are deformed with the perturbation of S_u. The restored property field of the de-faulted volume is then computed exactly as for restoration of a horizon in Steps (v) and (vi) in the procedure at the end of Sect. 4.4.2. The reverse process, that is, the inclusion of a fault in a volume, can be described and computed symmetrically.

Fig. 4.13 shows examples of how normal and reverse faults, with both normal and reverse drags (Grasemann *et al.*, 2005), can be modeled by applying different distribution functions to perturb the underlying T and S-fields of the parameterization. For example, normal drag is modeled by choosing $p_1 > 1$ and $p_2 > 1$ in the distribution function $d^t_{\texttt{bell}}(r_t)$ in the outward direction from the fault surface. This ensures natural deflection of layer boundaries in agreement with that of a normal drag. (See also Fig. 4.12(f).) In addition, $F_{\texttt{prop}}$, $\psi_{\texttt{adv}}$ and $\mathbf{a}$ are set so that the t-lines of the parameterization, with directions given by Eq. (4.4), are aligned with the layer boundaries adjacent to the fault surface. Reverse drags in the lower row of Fig. 4.13 were modeled by $p_1 = 1$ in $d^t_{\texttt{bell}}(r_t)$, which yields a distribution function that is convex without an inflection point.

Fig. 4.14(a) shows the result from a sequence of processes in a workflow including both faulting, erosion and folding. A new fault is activated in (b), and then the whole volume is again folded in (c). Fault displacements on each side of the fault surfaces were modeled independently with different influence areas and distribution functions.

Several extensions and generalizations can be introduced to model more complex displacement patterns, for example:

- Displacements other than the simple shear type in the u-direction can be modeled by vector displacement fields in the v- and t-directions. Displacement fields in the v-direction may simulate strike-slip faults, or oblique-slip faults when superimposed on the dip-slip displacement field. Displacement fields in the t-direction may simulate strain caused by compressional or extensional forces. Perturbations of S_v and T to form $\bar{S}_v$ and $\bar{T}$ are then performed similarly as for $\bar{S}_u$, and we get

$$\begin{aligned} \bar{S}_u &= S_u - \delta_u(u_e, v_e, 0) \cdot D_{\texttt{bell}}, \\ \bar{S}_v &= S_v - \delta_v(u_e, v_e, 0) \cdot D_{\texttt{bell}}, \quad \text{and} \\ \bar{T} &= T \ - \delta_t(u_e, v_e, 0) \cdot D_{\texttt{bell}}. \end{aligned} \tag{4.20}$$

- The influence region need not be restricted to an ellipsoid. A slightly more general shape is a superellipsoid. But, any shape that is convex, or more generally, star-shaped in the parameter space relative to the

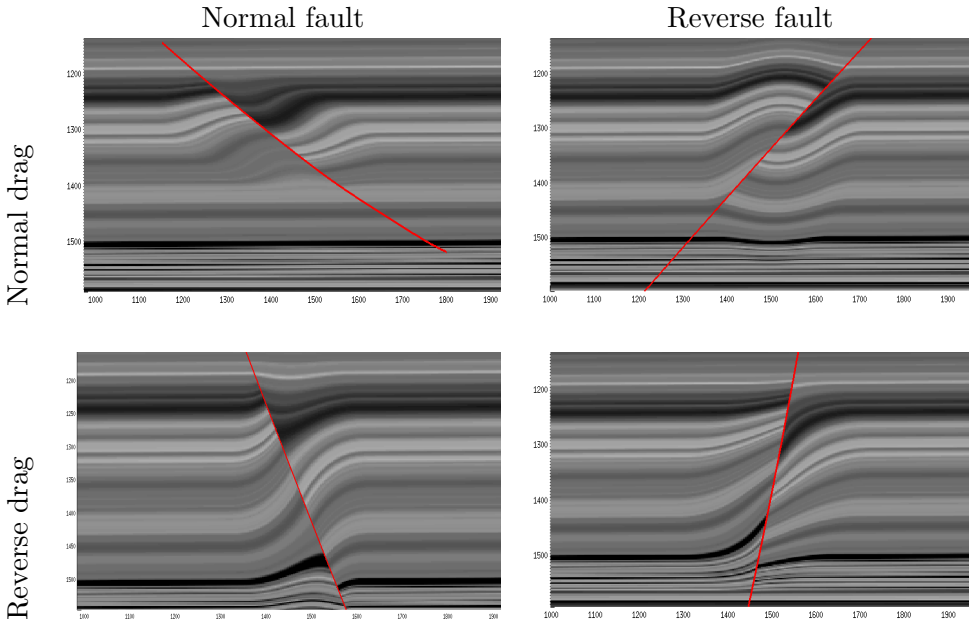


Fig. 4.13: Normal and reverse faults, with normal and reverse drags, modeled by using different distribution functions for perturbing the underlying T and S-fields of the parameterization. Normal drag and reverse drags are modeled by choosing $p_1 > 1$ and $p_1 = 1$, respectively, in the distribution function $d^t_{\texttt{bell}}(r_t)$ in the outward direction from the fault. (See also Fig. 4.12(f).)

center $(u_e, v_e, 0)$ could be used. Construction of such regions may require specialized interactive software modeling tools.

- Several displacement vectors, in addition to $\vec{\delta}_{\max}$, could be specified inside the influence region. The displacement field can then be constructed by interpolation, possibly with zero and zero derivative constraints at the boundary of the influence region to maintain a smooth distribution of displacements.

For the proposed fault modeling, the computationally most expensive operations are the generation of the T-field from the fault surface and the computation of the trivariate function from the scalar fields (S_u, S_v, T, Φ) in Step (v) of the procedure described at the end of Sect. 4.4.2. Since displacements take place inside the influence region only, these operations can be optimized to operate only on a subset of the regular grid. This enables an interactive work session for fault modeling where computations can be performed repeatedly with different parameters and thus approach a geologically feasible solution by trial and error. Note also that only a few parameters need to be considered

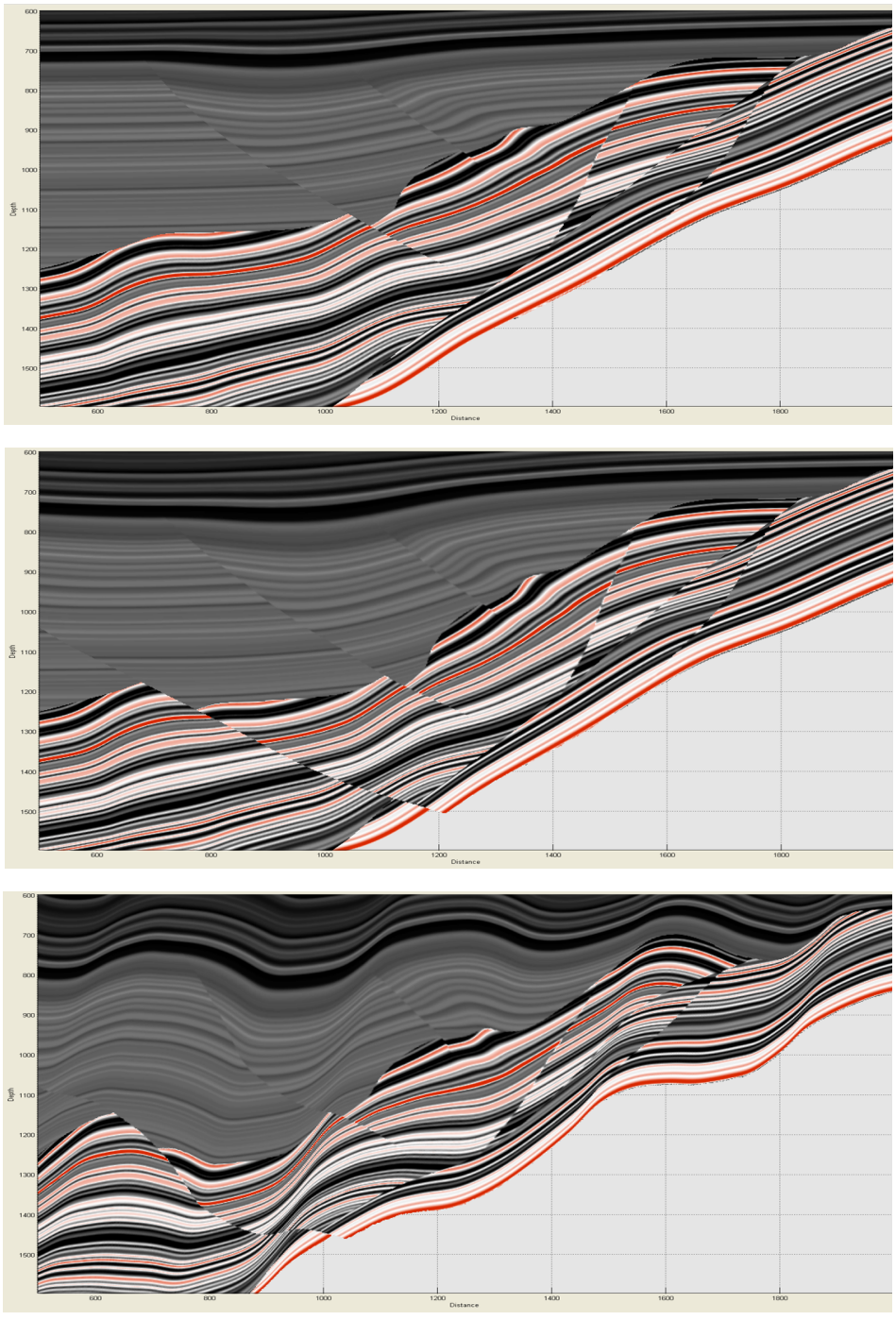


Fig. 4.14: (a): Faulted, eroded and folded slump. (b): A new fault is activated on the left with a displacement field that also intrudes the faults on the right. (c): Folding of the already deformed and faulted volume. (Screen shots from the mR-workflow in (CES).)

to model quite different deformation regimes. Other geological phenomena involving deformation of geological structures and displacement of material can probably also be modeled by vector displacement fields by using using the same principles and tools as described here. For example, extensional fault-propagation folding from the tip of a fault is an interesting topic for further research, as pointed out in Chapter 5.

4.5 Assessing Deformations

In this section, we derive useful metric properties from the parameterization of the 3D space and the scalar fields T, S_u and S_v defined on the finite difference grid of the geological volume. These metric properties can be used to quantify deformations related to folding, restoration and reconstruction. The mathematical framework we derive below is using fundamentals of classic differential geometry. The framework has similarities to Mallet (2004), but here we also establish the numerical framework for computing deformations and related quantities from T, S_u and S_v. We start by constructing a tangent space as a means for assessing strain and deformations in folded layers.

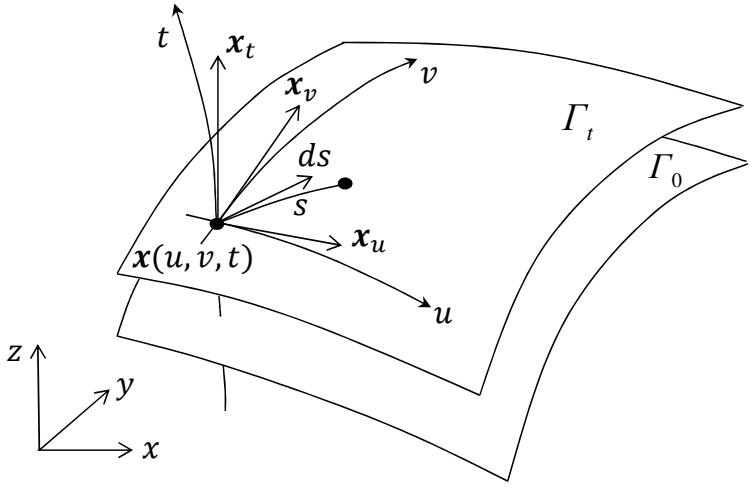


Fig. 4.15: Local frame and tangent space, and illustration of the first fundamental form on the isosurface Γ_t.

4.5.1 Construction of a Tangent Space

The parameterization introduced in Sect. 4.1 induces a local frame of vectors in each point $\mathbf{x}(u, v, t)$ in 3D space that we denote

$$\mathbf{x}_u = \frac{\partial \mathbf{x}}{\partial u}, \quad \mathbf{x}_v = \frac{\partial \mathbf{x}}{\partial v}, \quad \text{and} \quad \mathbf{x}_t = \frac{\partial \mathbf{x}}{\partial t},$$

as depicted in Fig. 4.15. These vectors are tangents to the u, v and t-lines of the parameterization, respectively. Note that since the t-lines are aligned with the group velocity vector in Eq. (4.4), we have $\mathbf{x}_t = \nabla_{\mathbf{p}} H$. The T-field may have corners and cusps with singularities in the derivatives (Sects. 2.3.1 and 3.1.5). Therefore, the derived functions $\mathbf{x}(u, v, t) = (\, x(u, v, t),\, y(u, v, t),\, z(u, v, t)\,)$ are not necessarily smooth everywhere. Still, in the following we will assume that the functions are smooth whenever this is needed for the discussion to ensure mathematical correctness. Moreover, we will also assume that $\mathbf{x}_u$, $\mathbf{x}_v$ and $\mathbf{x}_t$ are linearly independent such that the cross products $(\mathbf{x}_u \times \mathbf{x}_v)$, $(\mathbf{x}_v \times \mathbf{x}_t)$ and $(\mathbf{x}_t \times \mathbf{x}_u)$ are nonzero vectors. Furthermore, we assume that the three directions $\mathbf{x}_u$, $\mathbf{x}_v$ and $\mathbf{x}_t$ obey the right-hand rule like the vectors in Fig. 4.15. For a point $\mathbf{x}(u, v, 0)$ on the surface $\Gamma_0(u, v)$, $\mathbf{x}_u$ and $\mathbf{x}_v$ are tangent vectors to the surface, and more generally, for any point $\mathbf{x}$ the vectors $\mathbf{x}_u$ and $\mathbf{x}_v$ are tangents to an isosurface of the T-field through $\mathbf{x}$. Similarly, the vectors $\mathbf{x}_v$ and $\mathbf{x}_t$ are tangents to an isosurface of S_u, and $\mathbf{x}_t$ and $\mathbf{x}_v$ are tangent vectors to an isosurface of S_v. The normal vectors to isosurfaces S_u, S_v and T can then be expressed as

$$\begin{aligned} \mathbf{n}^u &= \frac{\nabla S_u}{\|\nabla S_u\|} = \frac{\mathbf{x}_v \times \mathbf{x}_t}{\|\mathbf{x}_v \times \mathbf{x}_t\|}, \\ \mathbf{n}^v &= \frac{\nabla S_v}{\|\nabla S_v\|} = \frac{\mathbf{x}_t \times \mathbf{x}_u}{\|\mathbf{x}_t \times \mathbf{x}_u\|}, \\ \mathbf{n}^t &= \frac{\nabla T}{\|\nabla T\|} = \frac{\mathbf{x}_u \times \mathbf{x}_v}{\|\mathbf{x}_u \times \mathbf{x}_v\|}. \end{aligned} \tag{4.21}$$

These normal vectors are well defined under the assumption of linear independency of $\mathbf{x}_u$, $\mathbf{x}_v$ and $\mathbf{x}_t$ stated above. It is also easy to establish the following orthogonality conditions

$$\begin{aligned} \nabla T \cdot \mathbf{x}_u = 0, \quad \nabla S_v \cdot \mathbf{x}_u = 0, \\ \nabla T \cdot \mathbf{x}_v = 0, \quad \nabla S_u \cdot \mathbf{x}_v = 0, \\ \nabla S_u \cdot \mathbf{x}_t = 0, \quad \nabla S_v \cdot \mathbf{x}_t = 0, \end{aligned}$$

where the two last conditions correspond to Eq. (4.5). Recall from Section 2.4.1 that $\nabla T \cdot \nabla_{\mathbf{p}} H = 1$. This also follows directly by differentiation along the

characteristic curve, where we have

$$\frac{dT}{ds} = \frac{1}{\|\mathbf{x}_t\|}$$

and

$$\frac{dT}{ds} = \nabla T \cdot \frac{\mathbf{x}_t}{\|\mathbf{x}_t\|},$$

thus giving $\nabla T \cdot \mathbf{x}_t = 1$. (Here, we have replaced $\nabla_{\mathbf{p}} H$ by $\mathbf{x}_t$.) It is straightforward to derive similar relations along the u- and v-parameter lines. Thus, we have

$$\begin{aligned} \nabla S_u \cdot \mathbf{x}_u &= 1, \\ \nabla S_v \cdot \mathbf{x}_v &= 1, \\ \nabla T \cdot \mathbf{x}_t &= 1. \end{aligned} \tag{4.22}$$

As a consequence of the orthogonality conditions in Eq. (4.5), the vector $\nabla S_u \times \nabla S_v$ points in the same direction as $\mathbf{x}_t$, which implies

$$\mathbf{x}_t = \|\mathbf{x}_t\| \cdot \frac{\nabla S_u \times \nabla S_v}{\|\nabla S_u \times \nabla S_v\|}.$$

If we take the scalar product with ∇T on both sides and rearrange, we get

$$\|\mathbf{x}_t\| = \frac{\|\nabla S_u \times \nabla S_v\|}{(\nabla S_u \times \nabla S_v) \cdot \nabla T},$$

and by combining the two expressions above we get

$$\mathbf{x}_t = \frac{\nabla S_u \times \nabla S_v}{(\nabla S_u \times \nabla S_v) \cdot \nabla T}. \tag{4.23}$$

Similarly, $\mathbf{x}_u$ and $\mathbf{x}_v$ can be derived in terms of ∇S_u, ∇S_v and ∇T

$$\mathbf{x}_u = \frac{\nabla S_v \times \nabla T}{(\nabla S_v \times \nabla T) \cdot \nabla S_u}, \tag{4.24}$$

$$\mathbf{x}_v = \frac{\nabla T \times \nabla S_u}{(\nabla T \times \nabla S_u) \cdot \nabla S_v}. \tag{4.25}$$

From an implementation point of view, we observe that the expressions for $\mathbf{x}_u$, $\mathbf{x}_v$ and $\mathbf{x}_t$ have a common denominator by using the formula $(\mathbf{a} \times \mathbf{b}) \cdot \mathbf{c} = (\mathbf{b} \times \mathbf{c}) \cdot \mathbf{a}$. The gradient fields ∇S_u and ∇S_v can be consistently computed for a grid point by the upwind construction in Eq. (4.6). This can be done along with the computation of T, ∇T, S_u and S_v in the front propagation algorithm such that the correct upwind construction is used. Alternatively, they can be

computed in a post-process directly from a dependency graph as explained in Sect. 4.2. The tangent vectors $\mathbf{x}_u$, $\mathbf{x}_v$ and $\mathbf{x}_t$ uniquely characterize the metric properties in the vicinity of a point $\mathbf{x}(u, v, t)$ and are used below for assessing deformation in folded layers. The vectors are computed in a grid point from grid point values of ∇S_u, ∇S_v and ∇T that have all been consistently computed from upwind values. The vector $\mathbf{x}_t$ can also be computed directly by Eq. (2.24) when ∇T is known, without using ∇S_u and ∇S_v.

Remark The two-dimensional equivalent to this tangent space is as follows: Let $\nabla S^{\perp}$ denote a vector field orthogonal to ∇S, with direction to the left as seen when looking in the direction of ∇S. Thus $\nabla S^{\perp} = (-S_y, S_x)$, and has the same direction as $\mathbf{x}_t$. Moreover, let $\nabla T^{-\perp} = (T_y, -T_x)$, that is, $\nabla T^{-\perp}$ is orthogonal to ∇T and pointing to the right as seen when looking in the gradient direction. Then

$$\mathbf{x}_u = \frac{\nabla T^{-\perp}}{\nabla T^{-\perp} \cdot \nabla S},$$
$$\mathbf{x}_t = \frac{\nabla S^{\perp}}{\nabla S^{\perp} \cdot \nabla T} \quad (= \nabla_{\mathbf{p}} H).$$

Similar to the 3D equivalent, we observe that the two tangent vectors have a common denominator.

□

4.5.2 Strain and the Metric Tensor

Many metric properties of a volume with folded, faulted and compacted layers can be characterized in the tangent space by the vectors $\mathbf{x}_u$, $\mathbf{x}_v$ and $\mathbf{x}_t$. The classical theory of differential geometry (Do Carmo, 1976; Kreyszig, 1991) uses the notion of the first fundamental form for surfaces to determine the intrinsic geometry in a neighbourhood of a given point. Consider, e.g., the arc s starting from $\mathbf{x}(u, v, t)$ on the isosurface Γ_t in Fig. 4.15 for which the parameter t is constant, and let ds be the principal linear part of the differential of s. Then the first fundamental form of Γ_t is

$$\begin{aligned} ds^2 &= (\mathbf{x}_u du + \mathbf{x}_v dv) \cdot (\mathbf{x}_u du + \mathbf{x}_v dv) \\ &= \mathbf{x}_u \cdot \mathbf{x}_u \, (du)^2 + 2\, \mathbf{x}_u \cdot \mathbf{x}_v \, (du\, dv) + \mathbf{x}_v \cdot \mathbf{x}_v \, (dv)^2. \end{aligned}$$

Thus, ds can be considered as the image on Γ_t of the displacement (du, dv) in a plane of the parameter space with constant t. The coefficients of this quadratic

form can be arranged in a matrix called a metric tensor

$$\mathbf{I} = (\mathbf{x}_u\ \mathbf{x}_v) \otimes (\mathbf{x}_u\ \mathbf{x}_v) = \begin{pmatrix} \mathbf{x}_u \cdot \mathbf{x}_u & \mathbf{x}_u \cdot \mathbf{x}_v \\ \mathbf{x}_u \cdot \mathbf{x}_v & \mathbf{x}_v \cdot \mathbf{x}_v, \end{pmatrix}.$$

where $\otimes$ denotes tensor product of vectors. This gives

$$ds^2 = (du\ dv)\ \mathbf{I} \begin{pmatrix} du \\ dv \end{pmatrix}.$$

Similarly, if we consider the displacement (du, dv, dt) in the parameter space, the image ds in the three-dimensional Euclidean space can be expressed by the quadratic form

$$\begin{aligned} ds^2 = &\mathbf{x}_u \cdot \mathbf{x}_u\,(du)^2 + \mathbf{x}_v \cdot \mathbf{x}_v\,(dv)^2 + \mathbf{x}_t \cdot \mathbf{x}_t\,(dt)^2 + \\ &2\mathbf{x}_u \cdot \mathbf{x}_v\,(du\,dv) + 2\mathbf{x}_u \cdot \mathbf{x}_t\,(du\,dt) + 2\mathbf{x}_v \cdot \mathbf{x}_t\,(dv\,dt). \end{aligned}$$

Then the coefficients can be arranged in a three-dimensional metric tensor

$$\mathbf{G}(\mathbf{x}) = (\mathbf{x}_u\ \mathbf{x}_v\ \mathbf{x}_t) \otimes (\mathbf{x}_u\ \mathbf{x}_v\ \mathbf{x}_t) = \begin{pmatrix} \mathbf{x}_u \cdot \mathbf{x}_u & \mathbf{x}_u \cdot \mathbf{x}_v & \mathbf{x}_u \cdot \mathbf{x}_t \\ \mathbf{x}_u \cdot \mathbf{x}_v & \mathbf{x}_v \cdot \mathbf{x}_v & \mathbf{x}_v \cdot \mathbf{x}_t \\ \mathbf{x}_u \cdot \mathbf{x}_t & \mathbf{x}_v \cdot \mathbf{x}_t & \mathbf{x}_t \cdot \mathbf{x}_t \end{pmatrix}, \quad (4.26)$$

such that

$$ds^2 = (du\ dv\ dt)\ \mathbf{G} \begin{pmatrix} du \\ dv \\ dt \end{pmatrix}.$$

The symmetric matrix $\mathbf{G}(\mathbf{x})$ characterizes the metric properties in the vicinity of a point $\mathbf{x}(u, v, t)$, and may serve as a tool to quantify and analyze spatial distribution of deformations. As shown below it can be used to characterize deformations during folding and compaction over geological time. In particular, $\mathbf{G}(\mathbf{x})$ can be used in applications for restoration and reconstruction to quantify deformations in the neighbourhood of the image of a point $\mathbf{x}$ in the restored state at the time of deposition. The entries in $\mathbf{G}(\mathbf{x})$ are easily computed on the finite difference grid from the equations in Sect. 4.5.1.

4.5.3 The Metric Tensor applied to Sedimentation, Compaction and Restoration

The definitions of sedimentation rate, compaction and decompacted sedimentation rate given below are taken from Mallet (2004). Assume that the folding is of Class 1B (parallel) type, such that sedimentary layers have constant orthogonal thickness along fold limbs. Consider a thin layer with observed

thickness Δh that has been deposited over a geological time lag Δt. Then the sedimentation rate is $\vartheta = \Delta h/\Delta t$ if there has not been any compaction. Assume that the T-field and the time variable t in the parameterized model (Sect. 4.1) correspond to geological time. The sedimentation rate at a point $\mathbf{x}$ can then be expressed as

$$\vartheta(\mathbf{x}) = \frac{1}{\|\nabla T(\mathbf{x})\|}.$$

Note that this equation corresponds to the isotropic case of the eikonal equation in our front propagation model with $\vartheta(\mathbf{x})$ taking the role as the normal propagation speed $F_{\texttt{prop}}(\mathbf{x})$ in Eq. (2.10). (See also the discussion in Sect. 2.3.1.) The sedimentation rate given above does not account for compaction of layers over geological time from the time of deposition. Therefore, the sedimentation rate calculated from observed layer thickness today is too slow if there has been compaction. Assume that compaction in a point is represented by a compaction coefficient $\phi(\mathbf{x}) \in [0, 1)$. The decompacted sedimentation rate is then

$$\vartheta_\phi(\mathbf{x}) = \frac{\vartheta(\mathbf{x})}{1 - \phi(\mathbf{x})}.$$

The actual geological time lag over which the layer has been deposited is

$$\Delta t(\mathbf{x}) = \frac{\Delta h_\phi(\mathbf{x})}{\vartheta_\phi(\mathbf{x})},$$

where $\Delta h_\phi(\mathbf{x})$ is the observed (decompacted) layer thickness at $\mathbf{x}$.

Consider a special case with a volume consisting of a stack of horizontal layers. Assume that the u-line and the v-line of the parameterization are aligned with the global x- and y-axes, respectively, and that an arc length parameterization is imposed such that tangent vectors $\mathbf{x}_u = (1, 0, 0)$ and $\mathbf{x}_v = (0, 1, 0)$. Assume further that the t-line of the parameterization is aligned with the vertical global z-axis and that t represents geological time. Then the tangent vector $\mathbf{x}_t$, which points upwards, can be interpreted as the sedimentation velocity

$$\mathbf{x}_t = (0, 0, \vartheta_\phi(\mathbf{x})).$$

Under these assumptions, for an unfolded layer, the metric tensor in Eq. (4.26) reduces to

$$\mathbf{G}_0(\mathbf{x}) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & |\vartheta_\phi(\mathbf{x})|^2 \end{pmatrix}.$$

Mallet (2004) proposed a restoration model to define a strain tensor characterizing the deformation of a layer between the time of deposition and present time. It is assumed that sediments are always deposited on a horizontal plane,

while at the time of deposition of a layer the deeper layers and horizons may have already been deformed. A thin layer between two horizons H_t and $H_{t-\Delta t}$ is considered, where H_t has been deposited at geological time t and the deeper horizon $H_{t-\Delta t}$ has been deposited at geological time $t - \Delta t$. The restoration process then transforms H_t to a horizontal layer as it was at the geological time t of deposition, while $H_{t-\Delta t}$ then was already deformed. Then the metric tensor defined above is used to characterize the metric properties in the neighbourhood of the image of a point $\mathbf{x}$ in the restored state at the time of deposition. The strain tensor characterizing the deformation of a layer between the time of deposition and present time is then defined as

$$\mathcal{E}(\mathbf{x}) = \frac{1}{2} \left(\mathbf{G}(\mathbf{x}) - \mathbf{G}_0(\mathbf{x}) \right).$$

More details on this mathematical model can be found in Mallet (2004) and the more recent Mallet (2008). As shown above, our meshless volumetric approach with scalar fields S_u, S_v and T consistently computed and defined on a regular grid, together with the derived tangent space in Sect. 4.5.1, provides all the tools needed to implement the metric tensors and quantify deformation locally in any point in space.

4.6 Summary

The parameterization of the 3D space proposed in this chapter is derived directly from the Hamilton-Jacobi framework for fold modeling presented in Chapter 2. This parameterization can be associated with a curvilinear grid in the Euclidean space consisting of directed u-lines, v-lines and t-lines as illustrated in Fig. 4.1. We derive a meshless representation of the parameterization on the finite difference grid where the T-field is computed. Thus, the parameterization is represented as a set of triples $\{(u, v, t)_{i,j,k}\}$ over the 3D grid. The t-values are taken from grid values of T. Grid values for u and v are computed in a finite difference scheme that propagates parameter values from the parametric surface $\Gamma_0(u, v)$, representing the reference layer, in the direction of the group velocity vector $\nabla_{\mathbf{p}} H$. We argue that this construction of the parameterization is a natural choice since isosurfaces of T are associated with a continuum of layer boundaries, and $\nabla_{\mathbf{p}} H$ points in the direction of the dip isogons of a fold.

This parameterization of the volume provides a powerful tool for restoration of layers and faults. The restoration scheme we present is meshless without the need for constructiong a complex tetrahedral mesh conforming to layer boundaries and fault surfaces. At the core of the restoration scheme there is

a mapping between a source and a target volume, both parameterized as outlined above. A blending scheme is proposed when layers are delimited by two or more boundaries. The underlying principle of the mapping is to preserve a particle's location (u, v, t) in the parameter space, while the particle is mapped to a new position (x, y, z) in the Euclidean space. A fast and stable mapping procedure is proposed that computes restored property values in the grid nodes of the restored volume. Restoration of faults follows a similar scheme. Then the fault surface represents both the source and the target geometry, and perturbation of parameter values in the parameter space is imposed to make a vector displacement field representing displacement of material induced by the fault.

The parameterization is also used to derive metric properties to quantify deformations related to folding and restoration of layers and faults. A tangent space is first derived, and then fundamentals of classic differential geometry is used to derive a metric tensor that characterizes metric properties in the volume. This is then applied to quantify spatial distribution of deformations related to restoration.

Chapter 5

Conclusions and Further Perspectives

The research reported in this thesis was motivated by the theoretical concept of Earth Recursion, and the process-based data restoration (dR) and model reconstruction (mR) workflows outlined in Chapter 1. The thinking behind Earth Recursion and the dR/mR-workflows is that by carefully examining the past and the geological evolution, we enhance our understanding of the present day geological environments. In particular, for the petroleum and mineral industries it is crucial to understand the processes that have acted through millions of years and created hydrocarbon and mineral reserves. The Hamilton-Jacobi mathematical and numerical frameworks presented in this thesis lay the foundation for implementation of processes in the proposed dR/mR-workflows. Most of the results in the preceding chapters, and the surface approximation method in Appendix A, have been realized as software libraries and implemented in CES. They have been used by engineers and geoscientists in Statoil ASA in a number of industrial cases, for example in the Troll field (Norway's largest oil field) and in prospects in the Middle East. Although the motivation for the research was the concept of Earth Recursion and the dR/mR-workflows, it should be emphasized that the results have wide applicability beyond the use in these contexts. Results from Chapters 2 and 3 have also been the starting point for other recent academic work (Gillberg *et al.*, 2012; Gillberg, 2013; Gillberg *et al.*, 2014).

In the following we first summarize conclusions from the technical chapters of this thesis, and then we point at some possible directions of further research based on the results.

Chapter 2 proposes a rigorous mathematical framework for modeling folds

in structural geology. A folded geological volume is modeled as a continuum of layer boundaries by a front propagation analogy and represented as a time of arrival field. The propagating front is governed by a static Hamilton-Jacobi equation with a velocity model consisting of two parts in general: a normal propagation speed and an advection speed in a specified direction. All five folding regimes classified in the classical literature (Ramsay, 1967) are then continuously spanned by varying the sign and magnitude of the two speed components. In the simplest isotropic case with zero advection, the Hamilton-Jacobi equation reduces to the eikonal equation, which models the parallel Class 1B fold. In the simplest anisotropic case, a linear PDE results, which models the Class 2 (similar) fold. The general anisotropic case consists of a generalized eikonal equation modeling the other fold classes: Class 1A, Class 1C and Class 3.

There is an exact match between the characteristic curves of the Hamilton-Jacobi equation and the dip isogons of the different folding classes. This constitutes a correspondence between the Hamilton-Jacobi mathematical framework and an intuitive geometrical representation that makes it easy to derive and analyze metric properties in the geological volume.

This novel approach to modeling folds differs from classical methods, which have mostly been based on geometric descriptions of the boundary surfaces of standardized folds. Our approach is Eulerian in the sense that the fold geometry is represented as a continuum in the whole 3D volume, and isosurfaces (isocurves in 2D) of the solution of the Hamilton-Jacobi equation can be regarded as a continuum of layer boundaries. Thus, metric properties such as distances, gradients (dip and strike), curvature, and their spatial variations are easily derived and represented as 3D continua covering the whole geological volume.

In Chapter 3 a robust numerical framework is derived to solve the Hamilton-Jacobi equation that models the different fold classes by the propagating front analogy. A variant of the fast marching method is introduced to compute the solution as a time of arrival field T on a regular grid. The Hamilton-Jacobi equation is discretized by upwind finite differences and a dynamic stencil construction to take anisotropy into account. While the original fast marching method for isotropic problems only uses stencil legs aligned with the grid lines, the dynamic stencil construction also uses diagonal directions both for better directional resolution and to satisfy the causality condition. Even though the neighbourhood used for updating a grid point is tentatively larger than for the original fast marching method, the computational complexity is retained as $O(N \log N_h)$, where N is the number of unknown grid points and N_h is the maximum number of grid values in the narrow band around the

propagating front Γ as it evolves in space and time. The basic structure of the original fast marching method is also maintained,

Based on the upwind finite difference discretization a local solver is derived consisting of a quadratic polynomial equation that is solved explicitly without iterations to find the solution value in a grid point. Once the solution values represented by the T-field are found, other metric properties can be consistently computed using the same upwind stencils as those used for computing the T-values.

Initialization of the boundary condition of the Hamilton-Jacobi equation is computed via a Lagrangian formulation and implemented by a Newton iteration scheme over smooth Bézier curves and surfaces representing the initial front Γ_0. This ensures high accuracy and robustness and avoids erroneous values to propagate to the computational domain where the solution values are computed with the dynamic upwind finite difference scheme. Smooth Bézier surfaces can be generated efficiently from autotrack data via the multilevel method presented in Appendix A. This method produces a semi-regular surface triangulation that can be resampled on a regular grid and interpolated by a bicubic tensor product Bézier surface that is used in our Newton iteration schemes.

Chapter 4 extends the mathematical and numerical frameworks from the two preceding chapters to support processes for data restoration, model reconstruction and property distribution. The parameterization of the 3D space introduced in this chapter forms the basis of a sound mathematical and numerical framework for restoration and property distribution, and for numerical schemes capable of quantifying deformations that take place through folding and faulting of a geological volume. The parameterization is consistently computed on the same finite difference grid as the T-field by exactly the same upwind construction. Thus, the representation format for all scalar fields representing the geological volume is meshless. Contrary to many other published methods for restoration, there is no need for constructing a complex tetrahedral mesh to honor surfaces of a structural model. Structural information representing curves and surfaces is only used initially when generating the parameterization of the 3D space, and the geometric representation is then kept separate from the computational model that performs the actual restoration. For example, unfolding horizons becomes conceptually simple by operating in the parameter space, and displacement of material by faulting and de-faulting are naturally imposed by perturbations of parameter values in the parameter space, such that displacements align naturally with the fault surface. This simplifies the numerical schemes for restoration of horizons and faults, and yields fast and stable numerical schemes and simple concepts for the user. Metric ten-

sors and deformations are also consistently quantified from standard concepts in differential geometry using the proposed parameterization as a reference system.

For certain deformation regimes, such as orthogonal flexure folding combined with superimposed homogeneous strain, the mathematical model reflects well the physical processes of fold formation, although geomechanical principles are not used directly in our framework. In particular, material lines deform with the t-lines of the parameterization, which ensures that the restoration model moves material to correct positions in the Euclidean space when the volume is mapped back in geological time.

Further Perspectives

There is a diversity of possible extensions and refinements of the mathematical and numerical frameworks presented in this thesis that can be useful in workflows in geomodeling software. A rigorous mathematical and numerical foundation has been established, which is easy to extend with new functionality. In the following we point at some directions of further research based directly on the results in the preceding chapters[1].

We have already demonstrated that finite strain states can be quantified and analyzed via the parameterization of the 3D space. We have also claimed that the mathematical model for restoration reflects the physical processes of fold formation, at least for certain deformation regimes, although geomechanical principles are not used in our restoration model. Further research could explore this further and compare the proposed scheme with methods based on finite element formulations to compute the restored model, and to quantify strain and stress states set up during fold formation (Maerten and Maerten, 2006; Moretti *et al.*, 2006).

In chapter 4 we associated the parameterization of the 3D space with a curvilinear grid in the Euclidean space. By the blending scheme we proposed in Sect. 4.3, this grid conforms naturally to the geological interfaces such that isosurfaces of the T-field coincide with the interfaces. It is tempting to utilize this logically rectangular grid for grid generation to produce grids for the finite element method, which is used in oil and gas reservoir simulators. This would have some similarities with a numerical technique proposed in Sethian (1999a), which is based on a propagating level set function. This technique produces grids that conform to one interface only, and only construction of 2D grids is

[1]Some of the ideas presented below have been prototyped in CES and other software, but we do not pursue these experiments in detail here since they are very premature.

demonstrated. Further directions of research indicated in Sethian (1999a) include generation of grids that confirm to several interfaces, extension to three dimensions, and blending with more traditional methods for grid generation. The Hamilton-Jacobi framework, with the meshless representation of the parameterization derived in chapter 4, may provide useful numerical tools to support research in these directions.

In Sect. 4.4.4 we demonstrate a powerful tool based on the Hamilton-Jacobi framework to model faults as vector displacement fields. The main ingredients in this setup are:

- a geometry (fault surface) with a parameterization used to control the space and to align the vector displacement field,
- distribution functions (bell shaped functions) in the three parameter directions used to distribute deformations around the geometry,
- an influence region used to limit the area where displacements take place, and
- perturbations of parameter values in the parameter space to model the actual deformations.

Displacement fields of, for example, normal and reverse faults with normal and reverse drags can then be modeled by choosing appropriate distribution functions, and T and S-fields for the parameterization. The resulting vector displacement field is represented meshless on the regular grid embedding the geological volume. We believe that similar principles can be applied to model other geological phenomena involving deformation of geological structures and displacement of material. For example, extensional fault-propagation folds have received attention in the oil industry due to the frequent occurrence of hydrocarbons associated with them (Hardy and McClay, 1999). Deformation patterns in connection to these foldings could be modeled starting from a parameterization of an extension of the fault surface from the fault tip. By choosing influence region, distribution functions, and T and S-fields for the parameterization carefully, realistic vector displacement fields could be generated to represent strain resulting from the extensional foldings. This would require the whole range of perturbations set up by Eq. (4.20). For example, one could aim at replicating strain patterns from the trishear kinematic models in Hardy and McClay (1999), and in Jin and Groshong Jr. (2006). Deformation of rock surrounding intrusive bodies like sills, dikes and salt domes, which are of great importance for hydrocarbon recovery (Braccini *et al.*, 2008; Petersen and Skjei, 2005), could also be modeled as vector displacement fields using the same principles.

Appendix A

Multilevel Least Squares Approximation of Scattered Data over Binary Triangulations[1]

An adaptive method for fitting surfaces to huge scattered data sets is presented. The approximation scheme generates multilevel triangulations obtained using a subdivision scheme known as longest edge bisection. Nested function spaces are defined over the multilevel triangulations. The approximation problem is solved by successive refinement of the triangulation while iterative methods are used for solving a system of linear equations at intermediate levels of the multilevel scheme. Regularization terms are coupled with a standard least squares formulation to guarantee uniqueness and control smoothness of the solution.

The motivation for the approximation scheme proposed here is to handle huge scattered data sets with noise, data with uneven distribution, and to handle situations where the surface topography varies rapidly over the domain. We emphasize the practical relevance of these qualities in applications like geological modeling, when fitting surfaces to data extracted with autotracking techniques, and approximation of cartographic data.

[1]This chapter is based on Hjelle and Dæhlen (2005) and Chapter 8 in Hjelle and Dæhlen (2006).

A.1 Related Work

Several multilevel methods for surface construction have been studied and presented over the past years, although relatively few deal with approximation of scattered data. In Lee *et al.* (1997) multilevel B-splines were used to generate a coarse-to-fine sequence of tensor product B-splines whose sum approaches the final approximation of given scattered data. The method is local in the sense that on the finer tensor product grids each B-spline coefficient is computed from nearby points only. Consequently, the method is also fast. However, if the scattered data are subject to noise or if the data are unevenly distributed over the domain, undesirable behaviour may occur near data locations. Numerical examples and explanations of these phenomena can be found in Lee *et al.* (1997), and in Hjelle (2001).

Hierarchical B-splines were introduced in Forsey and Bartels (1988), and applied to approximation of data arranged on a regular grid in Forsey and Bartels (1995). Application to scattered data was discussed in Greiner and Hormann (1997). Contrary to multilevel B-splines, the basic idea with hierarchical B-splines is to refine a tensor product grid locally where the approximation error exceeds a specified tolerance. Thereby a global approximation problem is subdivided into a series of local problems. Each local problem gives rise to a (small) linear equation system. This is done recursively until the B-spline surface approximates the scattered data within the given tolerance. The data structure is rather complex and implementation of the method requires considerable more effort than for multilevel B-splines.

A scheme for constructing smooth regular grid functions approximating scattered data was presented in Arge *et al.* (1995). In a first local step called regularization, a subset of the grid values are determined from nearby points using a classic scattered data interpolation technique like Shepard's method (Shepard, 1968) or the radial basis function method (Powell, 1992). Then, in the next extrapolation step, the grid values found in the first step are extended to the entire grid by solving a biharmonic differential equation. The method can also handle constraints (break lines) imposed on the surface, which makes the method suited for modeling faulted geological structures.

A.2 Binary Triangulations

An important part of the multilevel approximation scheme presented in this chapter is the construction of a nested sequence of semi-regular triangulations. We want to apply a subdivision scheme that generates an adaptive triangu-

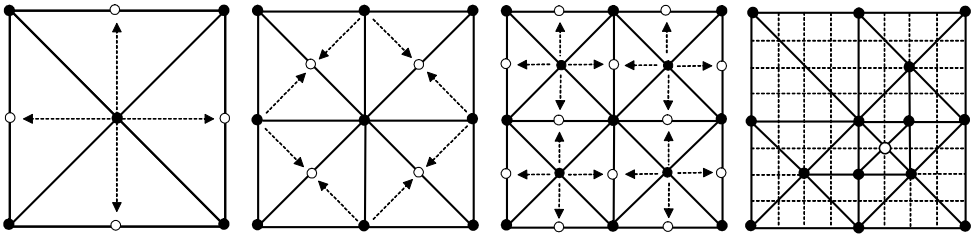


Fig. A.1: Recursive longest edge bisection. Dashed arrows indicate parent-child relationships. The rightmost triangulation has only one active vertex from the finest refine level.

lation where the triangle density reflects the variation in surface topography and distribution of the given data. A scheme known as longest edge bisection has become popular for view-dependent visualization (Lindstrom *et al.*, 1996; Röttger *et al.*, 1998). The scheme is also called $4-k$ meshes (Velho and Gomes, 2000), or restricted quad-tree triangulations (Pajarola, 1998). In the following, meshes generated by longest edge bisection are called binary triangulations since they can be considered as the result of recursive splitting of one triangle into two new triangles. One important property of the scheme, and variations over it, is that refinement can be done locally without the need to maintain the entire mesh at the same resolution.

Consider the initial triangulation on the left in Fig. A.1 with four isosceles triangles. The recursive splitting starts by inserting vertices on the longest edge of each triangle. To maintain a valid triangulation, new edges must be introduced as indicated by dashed arrows. The new triangulation has eight isosceles triangles, whose longest edges are bisected to obtain a new triangulation with 16 triangles (second from right). Each dashed arrow represents a parent-child relationship where the black bullets are parents and each circle represents a child. We say that a vertex is active when it has been inserted into the mesh. Every vertex at a certain refine level need not be active to obtain a valid triangulation. The general rule is that if a vertex is active, then its parents, its grandparents, etc. and all its ancestors at coarser levels must also be active. The four corner vertices are active by default. An example is shown by the rightmost binary triangulation in Fig. A.1. For the vertex drawn as a circle to be active, all the vertices drawn as black bullets must also be active. A unique triangulation then results with edges determined by the parent-child relationships involved.

We have implemented an algorithm based on the fact that the triangle vertices at any level of a binary triangulation constitute a subset of a regular grid. Starting from the initial triangulation on the left in Fig. A.1 we have a

grid Ψ_1 with 2×2 rectangular grid cells. For each grid cell in Ψ_1 a criterion for subdivision is examined. Criteria for subdivision will be explained shortly. If subdivision is required within the grid cell, the vertex in the middle of the cell, say v, belonging to the next finer grid Ψ_2 with 4×4 grid cells is activated. Then the parents of v must also be activated and a unique triangulation results with vertices which are a subset of those in the triangulation shown second from right. The algorithm proceeds on finer and finer grids where each grid Ψ_k is obtained by inserting grid lines halfway between the grid lines of Ψ_{k-1}.

We have implemented a simple data structure for the binary triangulation similar to that used in Lindstrom and Pascucci (2002) for view-dependent visualization. A triangulation is represented implicitly only by its (active) vertices and references from each vertex to its two parents. Algorithms operating on the data structure also become simple and efficient, for example algorithms for point localization and algorithms for generating triangle strips for efficient visualization.

A.3 Overview of the Approximation Scheme

The general scheme for computing surface triangulations over binary triangulations can briefly be outlined as follows. Given scattered data $P = \{(x_i, y_i)\}$ with corresponding real values $\{z_i\}$. Over each triangulation Δ_k in a coarse-to-fine sequence of binary triangulations we construct a least squares surface approximation f_k to the given data. Dependent on the application, different subdivision criteria can be used. For example, a prescribed tolerance may be given to control the maximum deviation between the surface approximation at the finest level and the given data values $\{z_i\}$. Thus, the accuracy of the approximation at one level can be used to decide where to refine, hence to produce the triangulation at the next finer level. For huge data sets it is important to reduce the amount of data or to create a triangulation that reflects the distribution of the scattered data. One way to achieve this, without taking any error measure into account, is to use a counting measure whereby Δ_k is refined as long as there is more than a prescribed number of data locations from P within some portion of the domain, for example inside a grid cell of Ψ_k. Variation over curvature measures could also be used to decide where to refine, for example by using a thin-plate energy measure locally over triangle edges. Since iterative methods are used for computing the least squares approximations, the solution found at one level can be used to compute a starting vector for the next level. More details on the numerical scheme and numerical examples are given in sections A.5 and A.6, respectively.

A.4 The Least Squares Approximation

We restrict the least squares approximation to piecewise linear surfaces. In geological modeling and geographic information systems systems, piecewise linear surfaces are often sufficient and may also be preferred to higher degree surfaces, which may cause undesirable oscillations (e.g. Gibbs phenomena) when modeling terrain with rapidly varying topography. Piecewise linear surfaces are also more efficient to compute and more compliant with other software components in such applications.

Given a set of distinct non-collinear data points

$$P = (x_1, y_1), (x_2, y_2), \ldots, (x_m, y_m),\ m \geq 3,$$

in the plane and corresponding real values $z_1, z_2, \ldots, z_m$, we seek a function f over the binary triangulation Δ that approximates the data. We restrict f to the finite dimensional space

$$S_1^0(\Delta) = \left\{ f \in C^0(\Omega) : f|_{t_i} \in \Pi_1 \right\},$$

where Π_1 is the space of bivariate linear polynomials, t_i is a triangle in Δ and Ω is a rectangular domain. As a basis for $S_1^0(\Delta)$ we use standard compactly supported nodal basis functions

$$N_1(x, y), N_2(x, y), \ldots, N_n(x, y)$$

which satisfy

$$N_i(v_j) = \delta_{ij},\ j = 1, \ldots, n,$$

where the $v_j = (x_j, y_j)$ are vertices in the underlying triangulation and δ_{ij} is the Kronecker delta function. Thus, a function in $S_1^0(\Delta)$ is written,

$$f(x, y) = \sum_{i=1}^{n} c_i N_i(x, y), \qquad c_i \in R.$$

Figure A.2 shows an example of a basis function $N_i(x, y)$. The support Ω_i of $N_i(x, y)$ is the union of those triangles that have $v_i = (x_i, y_i)$ as a common vertex. Each basis function is greater than zero strictly inside its support and zero elsewhere. The basis functions are also linearly independent[1] on Ω. This is an important (and necessary) property for the approximation schemes presented later in this chapter.

[1] Linear independency implies that for $\sum_{i=1}^{n} \alpha_i N_i(x, y)$ to be zero for all $(x, y) \in \Omega$, then all the real values α_i, $i = 1, \ldots, n$ must be zero.

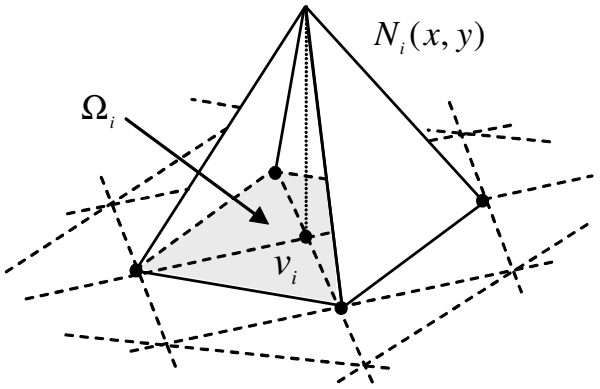


Fig. A.2: A basis function $N_i(x, y)$ for the function space $S_1^0(\Delta)$, and its compact support Ω_i.

In the basic least squares problem we seek a coefficient vector $\mathbf{c} = (c_1, \ldots, c_n)^T$ that minimizes

$$\sum_{k=1}^{m} (f(x_k, y_k) - z_k)^2 . \tag{A.1}$$

This classical least squares fitting problem always has a solution, however the solution is not necessarily unique. Also, the resulting surface $f(x, y)$ may not be sufficiently smooth, especially if the given data are subject to noise. We are particularly interested in different types of measured data, which often contain noise. One may augment Eq. (A.1) with a regularization term, also called smoothing term or penalty term in Von Golitschek and Schumaker (1990), to guarantee uniqueness and control smoothness of the solution. Many commonly used smoothing terms can be written on the quadratic form

$$\mathcal{J}(\mathbf{c}) = \mathbf{c}^T \mathbf{E} \mathbf{c}, \tag{A.2}$$

where $\mathbf{E}$ is a symmetric and positive semidefinite $n \times n$ matrix. When working with spaces of functions of higher degree, the smoothing term usually involves first and second derivatives. For the piecewise linear space $S_1^0(\Delta)$ we may use discrete analogues where $\mathcal{J}(\mathbf{c})$ is the sum of some roughness measure around each vertex or across each interior edge of the triangulation. The functional we want to minimize is now

$$I(\mathbf{c}) = \sum_{k=1}^{m} (f(x_k, y_k) - z_k)^2 + \lambda \mathbf{c}^T \mathbf{E} \mathbf{c} = \|\mathbf{B}\mathbf{c} - \mathbf{z}\|^2 + \lambda \mathbf{c}^T \mathbf{E} \mathbf{c}, \tag{A.3}$$

for some real value $\lambda \geq 0$, and where $\| \cdot \|$ denotes the Euclidean norm. Here

$\mathbf{z} = (z_1, \ldots, z_m)^T$ and $\mathbf{B}$ is the $m \times n$ matrix

$$\mathbf{B} = \begin{pmatrix} N_1(x_1, y_1) & \cdots & N_n(x_1, y_1) \\ N_1(x_2, y_2) & \cdots & N_n(x_2, y_2) \\ \vdots & & \vdots \\ N_1(x_m, y_m) & \cdots & N_n(x_m, y_m) \end{pmatrix}. \tag{A.4}$$

Setting the gradient of $I(\mathbf{c})$ equal to zero leads to the normal equations

$$\left(\mathbf{B}^T\mathbf{B} + \lambda\mathbf{E}\right)\mathbf{c} = \mathbf{B}^T\mathbf{z}. \tag{A.5}$$

The $n \times n$ matrix $\mathbf{B}^T\mathbf{B}$ is symmetric and positive semidefinite, so the solution to Eq. (A.5) with $\lambda = 0$ is not necessarily unique. If $\mathbf{E}$ is positive definite, then the system matrix $\left(\mathbf{B}^T\mathbf{B} + \lambda\mathbf{E}\right)$ with $\lambda > 0$ is also positive definite, and thereby nonsingular, which implies that Eq. (A.5) has a unique solution.

To justify the use of the smoothing term for controlling the behaviour of the least square fit, we briefly go through a result in Von Golitschek and Schumaker (1990). Assume for now that $\left(\mathbf{B}^T\mathbf{B} + \lambda\mathbf{E}\right)$ is nonsingular for $\lambda \geq 0$. Let $f_\lambda(x, y)$ be the penalized least squares fit to the given data with the smoothing parameter λ. Then the mean square error given by

$$T_{\mathbf{z}}(\lambda) = \frac{1}{m}\sum_{k=1}^{m}\left(f_\lambda(x_k, y_k) - z_k\right)^2 \tag{A.6}$$

is monotone increasing for $\lambda \geq 0$ with derivatives $T'_{\mathbf{z}}(0) = 0$ and $\lim_{\lambda\to\infty} T'_{\mathbf{z}}(\lambda) = 0$. Thus, λ controls the trade-off between smoothness and mean square error. A best fit, best in the sense of minimizing the mean square error, is then obtained by setting the value of the smoothing parameter λ to zero.

It can be shown that the basic least squares problem $\mathbf{B}^T\mathbf{B}\mathbf{c} = \mathbf{B}^T\mathbf{z}$ without a smoothing term has a unique solution if and only if $\mathbf{B}$ has linearly independent columns (Golub and Loan, 1996). If one or more columns of $\mathbf{B}$ are zero vectors, then the columns are not linearly independent. We observe from Eq. (A.4) that each basis function $N_j,\ j = 1, \ldots, n$ makes a column. A column, say column number l, is a zero vector if $N_l(x_i, y_i) = 0$ for $i = 1, \ldots, m$. This happens if no point of P falls strictly inside the domain Ω_l of N_l, which often occurs for cartographic data and geological data, whose distribution may vary rapidly over the domain. For instance, autotrack data and hypsographic data (contour lines) in cartography typically possess such uneven distributions. So in most cases when constructing surfaces from such data, we need a smoothing parameter λ greater than zero to guarantee uniqueness of Eq. (A.5).

An approximation scheme without a smoothing term based on refinement of arbitrary triangulations was proposed by Rippa (1992). Uniqueness then relied on the fact that the vertices of the triangulation were always chosen as a subset of the input data. Consequently, there are n rows of $\mathbf{B}$ that constitute the identity matrix (with a proper ordering of the rows), and the columns are trivially linearly independent.

The system matrix $\mathbf{B}^T\mathbf{B}$ of the basic least squares problem is clearly sparse. Indeed, an element

$$\left(\mathbf{B}^T\mathbf{B}\right)_{ij} = \sum_{k=1}^{m} N_i(x_k, y_k) N_j(x_k, y_k)$$

is non-zero only if $i = j$ or if the domains of the basis functions N_i and N_j overlap with two triangles that share a common edge. Thus, a non-zero off-diagonal element $(\mathbf{B}^T\mathbf{B})_{ij}$ corresponds to an edge in the triangulation connecting the two vertices v_i and v_j. It can be shown that the number of edges in a triangulation with V vertices has an upper bound $3V - 6$. Then the number of non-zero off-diagonal elements in the $n \times n$ matrix $\mathbf{B}^T\mathbf{B}$ has an upper bound $2(3n - 6)$, and counting the diagonal which is also non-zero, we find that the average number of non-zeros in each row is approximately 7.

A.4.1 Smoothing Terms

Smoothing terms on the quadratic form in Eq. (A.2) can be obtained from the membrane energy,

$$\int |\nabla g|^2 = \int g_x^2 + g_y^2, \tag{A.7}$$

and from an approximation of the thin-plate energy,

$$\int g_{xx}^2 + 2g_{xy}^2 + g_{yy}^2, \tag{A.8}$$

for some differentiable function g. Loosely speaking, in the context om minimizing the functional (A.3), the membrane energy prefers surfaces with small area, while the thin-plate energy prefers surfaces with small overall curvature. Since functions in $S_1^0(\Delta)$ are not twice differentiable, we will use an approximation to the thin-plate energy based on divided differences.

The smoothing term based on the membrane energy can be expressed on

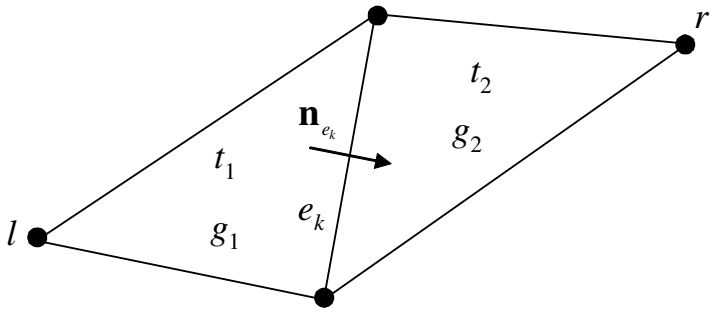


Fig. A.3: Stencil for the second order divided difference operator

the following quadratic form,

$$\begin{aligned}\mathcal{J}_1(\mathbf{c}) &= \int |\nabla f|^2 = \int \left| \sum_{i=1}^{n} c_i \nabla N_i \right|^2 \\ &= \sum_{i=1}^{n} \sum_{j=1}^{n} \int (\nabla N_i \cdot \nabla N_j)\, c_i c_j = \mathbf{c}^T \mathbf{E} \mathbf{c}.\end{aligned}$$

Thus, the elements of the smoothing matrix are

$$E_{ij} = \int \nabla N_i \cdot \nabla N_j.$$

Matrix $\mathbf{E}$ is symmetric and sparse with the same sparsity pattern as $\mathbf{B}^T\mathbf{B}$, thus E_{ij} is non-zero when $i = j$ or when (i, j) corresponds to an edge in the triangulation.

An approximation to the thin-plate energy functional in Eq. (A.8) can be based on a second order divided difference operator. Fig. A.3 shows a stencil with the triangles and vertices involved in the construction of the divided difference. Let $g_i = f|_{t_i},\ i = 1, 2$, where t_1 and t_2 are the triangles sharing the edge e_k. Further, let $\mathbf{n}_{e_k}$ be a unit vector in the xy-plane orthogonal to the projection of e_k in the xy-plane, and let $|E_I|$ be the number of interior edges in the triangulation. We define the discrete thin-plate energy as the following sum over all interior edges,

$$\mathcal{J}_2(\mathbf{c}) = \sum_{k=1}^{|E_I|} \int_{e_k} [\nabla f \cdot \mathbf{n}_{e_k}]_J^2$$

where $[\nabla f \cdot \mathbf{n}_{e_k}]_J = (\nabla g_2 - \nabla g_1) \cdot \mathbf{n}_{e_k}$ is the jump in the derivative of f over the interior edge e_k in the direction of $\mathbf{n}_{e_k}$. This measure was used in Dyn

et al. (1990) for constructing data dependent triangulations, and equivalent measures were used in Guskov (1998), and in Guskov *et al.* (1999) as divided difference operators in subdivision schemes for triangulations. We obtain the following quadratic form,

$$\begin{aligned}\mathcal{J}_2(\mathbf{c}) &= \sum_{k=1}^{|E_I|} \int_{e_k} \left[\sum_{i=1}^{n} c_i (\nabla N_i \cdot \mathbf{n}_{e_k}) \right]_J^2 \\ &= \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{|E_I|} \int_{e_k} \left[\nabla N_i \cdot \mathbf{n}_{e_k}\right]_J \left[\nabla N_j \cdot \mathbf{n}_{e_k}\right]_J c_i c_j = \mathbf{c}^T \mathbf{E} \mathbf{c}\end{aligned}$$

where

$$E_{ij} = \sum_{k=1}^{|E_I|} \int_{e_k} \left[\nabla N_i \cdot \mathbf{n}_{e_k}\right]_J \left[\nabla N_j \cdot \mathbf{n}_{e_k}\right]_J .$$

In general, an element E_{ij} is non-zero if $i = j$, or if i and j correspond to vertices of the same stencil. This occurs if the line segment (v_i, v_j) spans an edge, or if v_i is the vertex on the opposite side of an edge from v_j. The latter case corresponds to the vertices with indices l and r in Fig. A.3. Compared to the operator for the membrane energy, this operator generates a sparsity pattern with more non-zero entries in the system matrix.

A.4.2 Uniqueness

It follows from basic linear algebra that since both matrices $\mathbf{B}^T\mathbf{B}$ and $\mathbf{E}$ are positive semidefinite, the system matrix $\left(\mathbf{B}^T\mathbf{B} + \lambda\mathbf{E}\right)$ for $\lambda > 0$ can also be positive semidefinite. But here we show through geometric analysis that the system matrix is strictly positive definite such that the system of equations in (A.5) has a unique solution.

For $(\mathbf{B}^T\mathbf{B} + \lambda\mathbf{E})$ to be positive semidefinite, we must have

$$\mathbf{c}^T \left(\mathbf{B}^T\mathbf{B} + \lambda\mathbf{E}\right) \mathbf{c} = \mathbf{c}^T \left(\mathbf{B}^T\mathbf{B}\right) \mathbf{c} + \lambda \mathbf{c}^T \mathbf{E} \mathbf{c} = 0, \tag{A.9}$$

where both terms $\mathbf{c}^T(\mathbf{B}^T\mathbf{B})\mathbf{c} = 0$ and $\mathbf{c}^T\mathbf{E}\mathbf{c} = 0$ for some vector $\mathbf{c} \neq \mathbf{0}$. We first observe that $\mathbf{c}^T\mathbf{E}\mathbf{c} = \mathcal{J}(\mathbf{c})$, the general energy term.

The membrane energy generates the functional

$$\mathcal{J}_1(\mathbf{c}) = \sum_{k=1}^{|T|} A_k \left|\nabla g_k\right|^2 = \sum_{k=1}^{|T|} A_k \left[(\partial g_k/\partial x)^2 + (\partial g_k/\partial y)^2 \right],$$

where $|T|$ is the number of triangles and A_k is the area of triangle t_k (in the xy-plane). For this expression to be zero we must have $\partial g_k/\partial x = 0$ and $\partial g_k/\partial y = 0$ for $k = 1, \ldots, |T|$. But then, since the triangulation is connected, all coefficients must be equal. Thus, $\mathbf{c}^T\mathbf{E}\mathbf{c} = 0$ if and only if $c_1 = c_2 = \cdots = c_n$. Furthermore, for the first term in Eq. (A.9) stemming from the basic least squares problem, we must have

$$\mathbf{c}^T(\mathbf{B}^T\mathbf{B})\mathbf{c} = \|\mathbf{B}\mathbf{c}\|^2 = 0, \tag{A.10}$$

which implies that $(\mathbf{B}\mathbf{c})_j = \sum_{i=1}^{n} c_i N_i(x_j, y_j) = f(x_j, y_j) = 0$ for all $j = 1, \ldots, m$. But since all coefficients c_k are equal, the function f must then be zero everywhere and all coefficients c_k must be zero. Thus, $\mathbf{c}^T(\mathbf{B}^T\mathbf{B} + \lambda\mathbf{E})\mathbf{c} = 0$ implies that $\mathbf{c} = \mathbf{0}$ and we conclude that the system matrix $(\mathbf{B}^T\mathbf{B} + \lambda\mathbf{E})$ is indeed positive definite, and therefore nonsingular. Hence, the system of equations (A.5) always has a unique solution when $\mathbf{E}$ is constructed from the membrane energy.

The discrete thin-plate operator generates the functional

$$\mathcal{J}_2(\mathbf{c}) = \sum_{k=1}^{|E_I|} \int_{e_k} \left[(\nabla g_2 - \nabla g_1) \cdot \mathbf{n}_{e_k}\right]^2 .$$

For this expression to be 0 for $\mathbf{c} \neq \mathbf{0}$, and thus for $\mathbf{E}$ to be positive semidefinite, we must have $(\nabla g_2 - \nabla g_1) = \mathbf{0}$ over all interior edges in the triangulation. Then, since the triangulation is connected, all the gradients of $f|_{t_i}$ over all triangles t_i in Δ are equal and f must be a linear polynomial. But then Eq. (A.10) again implies that $f(x_j, y_j) = 0$ for $j = 1, \ldots, m$ under the same assumption that $\mathbf{B}^T\mathbf{B}$ is positive semidefinite. This implies, that f is zero everywhere and that all coefficients in $\mathbf{c}$ must be zero. Thus, we arrive at the same conclusion as above that the system matrix $(\mathbf{B}^T\mathbf{B} + \lambda\mathbf{E})$ is positive definite and that Eq. (A.5) always has a unique solution when $\mathbf{E}$ is constructed from the thin-plate energy functional.

A.5 Numerical Schemes

At each level k of the multilevel scheme, the system of equations in (A.5) is relaxed with the Gauss-Seidel method (or alternatively the conjugate gradient method). A nested iteration scheme has been implemented where the resulting surface $f_k \in S_1^0(\Delta_k)$ at one level is used to supply an initial guess for $f_{k+1} \in S_1^0(\Delta_{k+1})$ at the next level. Since all vertices of the triangulation Δ_k are also in Δ_{k+1}, all coefficients from f_k are transferred directly to the next level as starting values for the unknown coefficients of f_{k+1}. Moreover, each coefficient

c_{ij}^{k+1} of f_{k+1} corresponding to a vertex (x_i, y_j) in Δ_{k+1} that is not in Δ_k, is simply initialized as $c_{ij}^{k+1} = f_k(x_i, y_j)$. Since the triangulations generated by the binary subdivision scheme are nested, the function spaces defined over them constitute a nested sequence of subspaces, $S_1^0(\Delta_1) \subset S_1^0(\Delta_2) \subset \cdots \subset S_1^0(\Delta_h)$, where Δ_h is the triangulation at the finest level. In addition to providing fast solution of the linear equation system, this coarse-to-fine scheme also generates a sequence of surface approximations to the scattered data at different levels of detail.

In Von Golitschek and Schumaker (1990) generalized cross validation is proposed for computing the smoothing parameter λ. This method is rather CPU-extensive. In our implementation we use a simpler approach adopted from Floater (1998), where the default value is set to

$$\lambda_d = \left\|\mathbf{B}^T\mathbf{B}\right\|_F / \left\|\mathbf{E}\right\|_F , \tag{A.11}$$

where $\|\cdot\|_F$ denotes the Frobenius matrix norm. The idea is that the contributions from $\mathbf{B}^T\mathbf{B}$ and $\lambda\mathbf{E}$ to the system matrix in Eq. (A.5) should have roughly the same weight. For data without noise this value works well in most cases, but with noise present a much larger λ, up to 1000 times λ_d, is necessary to obtain a sufficiently smooth solution.

At the first levels of the multilevel scheme the system is solved to yield an exact solution. This establishes a good global trend of the surface as a basis for successive improvements when iterating at the finer levels with more unknowns. A combined stopping criterion based on relative improvement of the solution and decrease of the residual, measured by the l_2 norm, was used for the iterative solver. The Gauss-Seidel method performed better than the conjugate gradient method as an iterative solver, even though the conjugate gradient method converges faster when applied on a fixed mesh without a good initial guess. In most cases, between 10 and 20 iterations at each level were sufficient. Any further iterations did not improve the solution significantly when the smoothing parameter was chosen as in Eq. (A.11). But with larger λ the number of iterations at each level was higher, e.g. up to 200 when λ was between 500 and 1000 times λ_d.

A natural improvement of this simple coarse-to-fine ascending scheme is to include recursive coarse grid correction at each level and thus obtain a true geometric multigrid solver (Briggs *et al.*, 2000). To ease implementation we used a standard algebraic multigrid solver from the software library ML (Hu *et al.*, 2000), by which the coarse grid correction step need not be provided explicitly by the user. This library is generic in the sense that its algorithms can operate on any data structure for vectors and matrices if a specified interface is provided by the user. Both full multigrid and repeatedly V-cycles and W-

cycles are available in ML. The algorithms, which takes as input a system of equations at the finest level only, run with a fixed number of iterations at each level specified by the user, for example between five and ten Gauss-Seidel iterations as in our examples. For details on the theory of algebraic multigrid, see for example Brandt *et al.* (1984) and Brandt (1986).

With the default smoothing parameter in Eq. (A.11), the algebraic multigrid schemes and the simple coarse-to-fine scheme outlined above were approximately equally good. But for larger λ the multigrid schemes were significantly faster. An advantage with multigrid, if V-cycles or W-cycles are employed, is that an initial guess for the solution at the finest level can be given when starting the solver. A good approximation of the coefficient in each triangle vertex is easily obtained in most cases by a fast local approximant or interpolant that uses nearby scattered data only, for example Shepard's method (Shepard, 1968).

Due to an ill-condition system there is no clear correspondence between the residual and the error, and thus, the number of repeated V-cycles or W-cycles might be difficult to determine. A solution in practical applications is to examine by visual inspection if the resulting surface triangulation is sufficiently smooth and pleasant looking, and then decide if additional multigrid cycles should be performed.

A.6 Numerical Examples

In this section we present numerical examples based on two different data sources: data sampled from Franke's test function (Franke, 1982) and real data from a terrain consisting of a combination of hypsographic data (contour data) and scattered measurements from the terrain surface. The regularization term is based on the thin-plate energy in both examples.

Approximation of data sampled from Franke's function. (Consult Figs. A.4 and A.5). The scattered data set consists of 3000 points sampled over the unit square. The points are unevenly distributed in the domain with relatively more data in areas with steep gradient or high curvature. A combined subdivision criterion based on a counting measure and an error measure is used. A grid cell is thus refined if more than two points are inside the grid cell and the error for at least one of the points inside the cell is greater than the prescribed tolerance. The tolerance was 0.25 percent of $|z_{\max} - z_{\min}|$ of the given data, and the default value for λ was used. The coarse-to-fine algorithm terminated after subdivision of the grid Ψ_7 and delivered a triangulation Δ_8

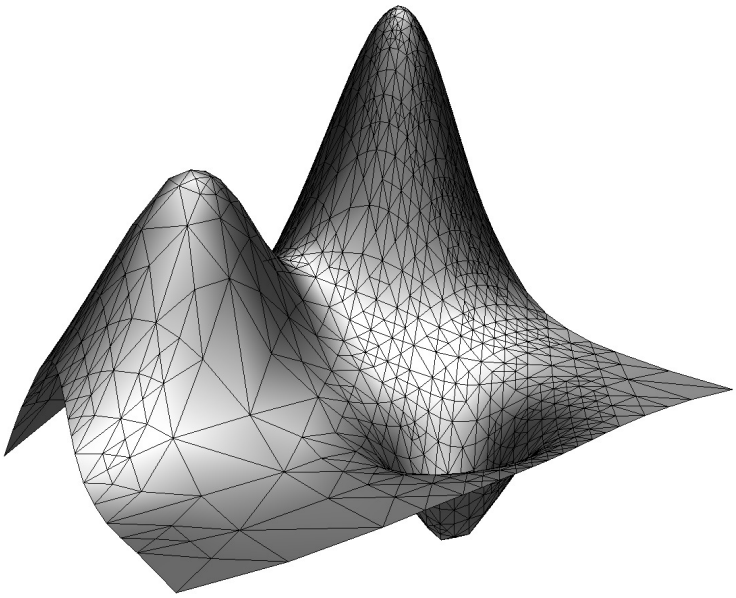


Fig. A.4: Approximation to the Franke function, and the resulting binary triangulation imposed on the surface

with 1476 vertices and 2866 triangles. The number of Gauss-Seidel iterations at each level was between 9 and 12 to reach the stopping criterion. As expected, there are more triangles in areas with large curvature and high density of data due to the combined error and counting measure used as a subdivision criterion. Also note the nice spatial grading from small triangles to larger triangles in Fig. A.5. If the middle vertex of all grid cells were activated when operating on Ψ_7, the resulting triangulation would have 33025 vertices, which also would be the number of unknowns in the equation system at that level. Thus, less than 4.5 percent of the maximum number of available vertices are used in the triangulation.

Fig. A.6 demonstrates the effect of choosing a very large λ (10 000 times λ_d), and thus demanding much smoothing. The surface leaves the given data points, and for even larger λ when the smoothing term becomes more dominant, the surface approaches a plane. Also recall that the mean square error given by Eq. (A.6) increases monotonically towards a maximum with increasing λ.

Approximation of noisy data from Franke's function. Normally distributed noise was added to the data set used in the previous example. Subdivision of a grid cell was performed when there was more than two points inside

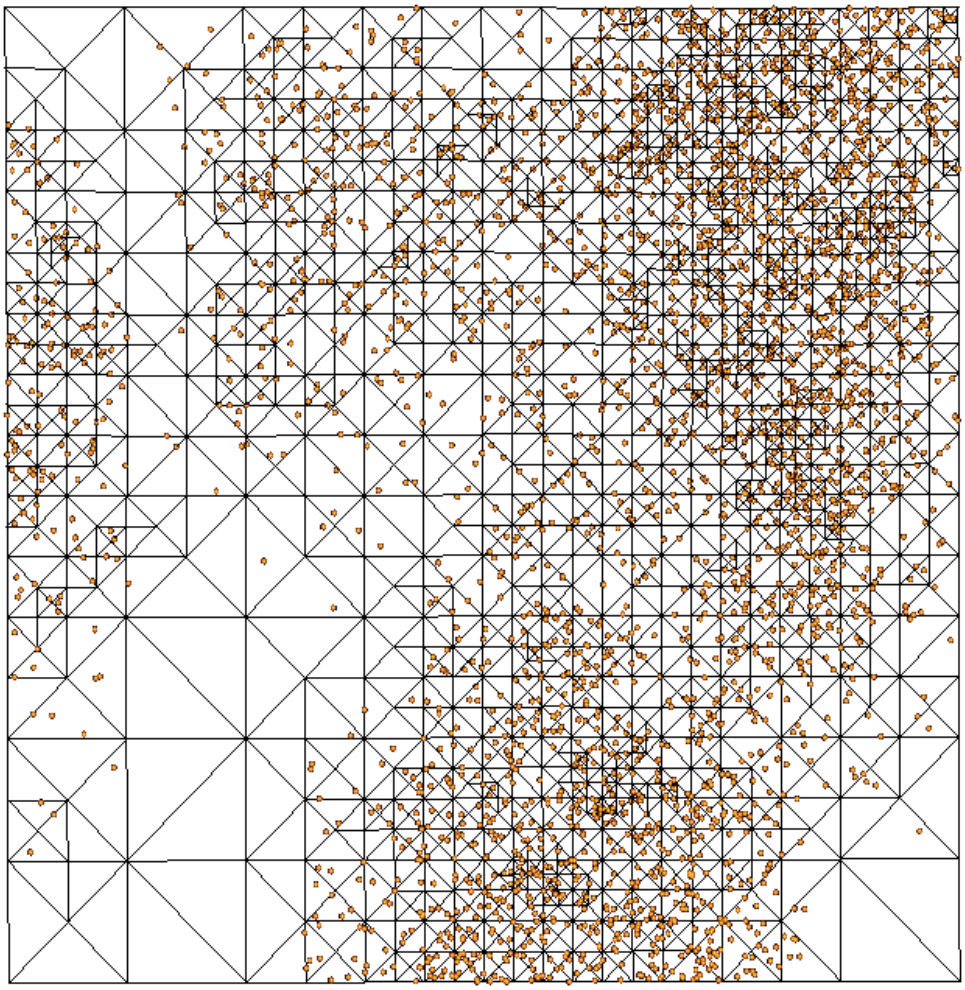


Fig. A.5: The same binary triangulation as in Fig. A.4 together with input data for numerical examples. The lower left corner corresponds to the nearest corner in Fig. A.4.

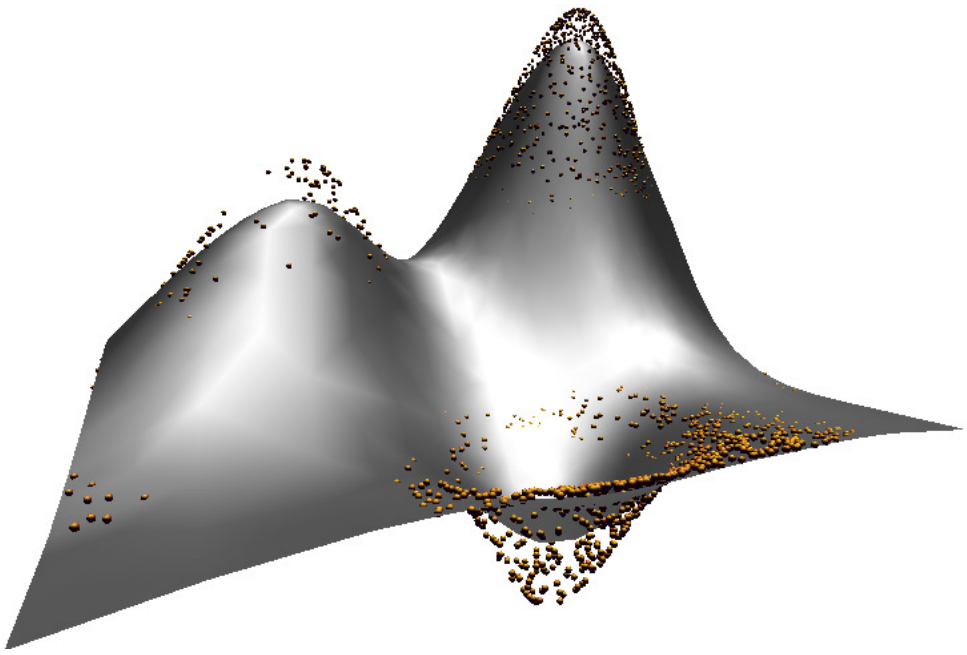


Fig. A.6: Approximation with huge smoothing parameter

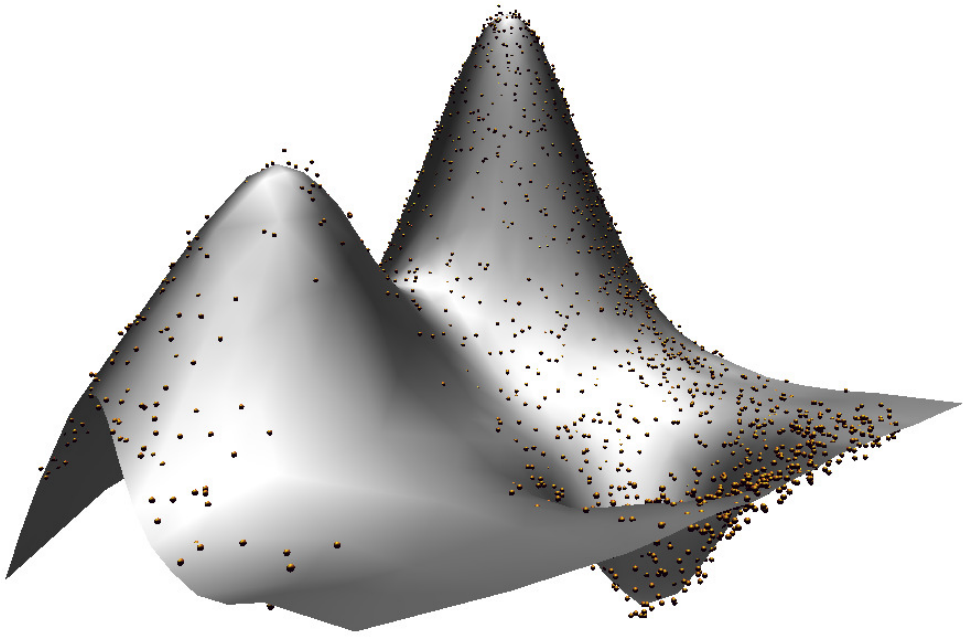


Fig. A.7: Approximation to Franke's function from a data set with noise

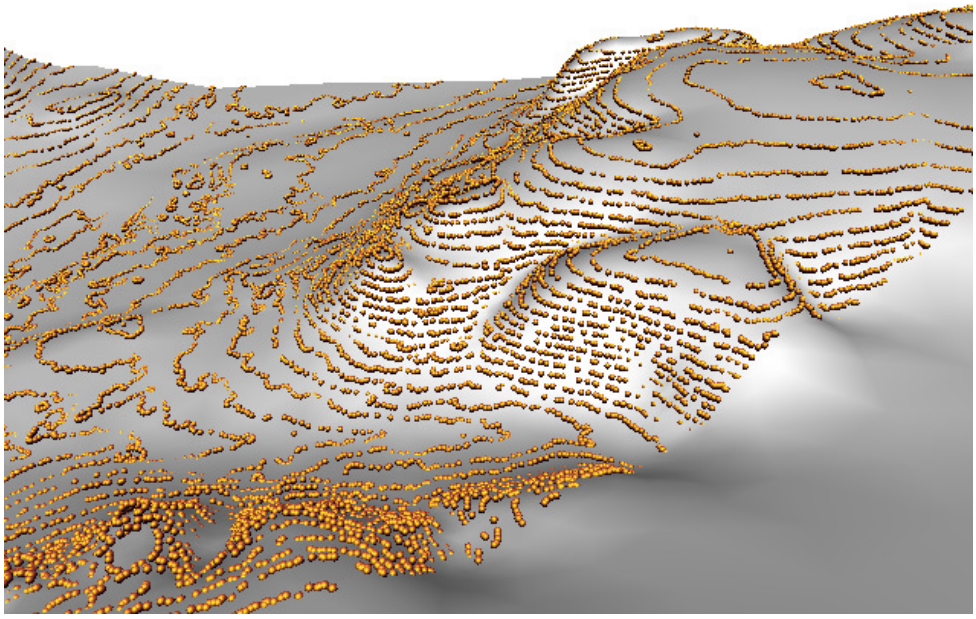


Fig. A.8: Approximation of terrain data, and the given data imposed on the surface

the cell, but no error measure was used. To obtain a smooth pleasant looking surface comparable to the surface produced in the previous example, it was necessary to increase λ to 600 times λ_d. The number of Gauss-Seidel iterations at each level with the coarse-to-fine scheme was between 73 and 207. The algorithm terminated at the same level as in the previous example. Fig. A.7 shows the approximation and the given noisy data points.

When algebraic multigrid was used starting from the triangulation produced at the finest level, 5 V-cycles with 6 Gauss-Seidel iterations at each level were necessary to obtain the same residual norm. (The initial guess was simply the mean value of all data values $\{z_i\}$.) When more noise was added to the data and λ was increased, multigrid was even more superior.

Terrain modeling from hypsographic data and scattered data. The terrain model shown in figures A.8 and A.9 was derived from approximately 50 000 points consisting of both hypsographic data and scattered data points measured from the underlying terrain. An error measure was used as a subdivision criterion although the data contained noise. An acceptable smooth surface was obtained with a smoothing factor 10 times λ_d. The number of Gauss-Seidel iterations at each level was between 41 and 593 by the

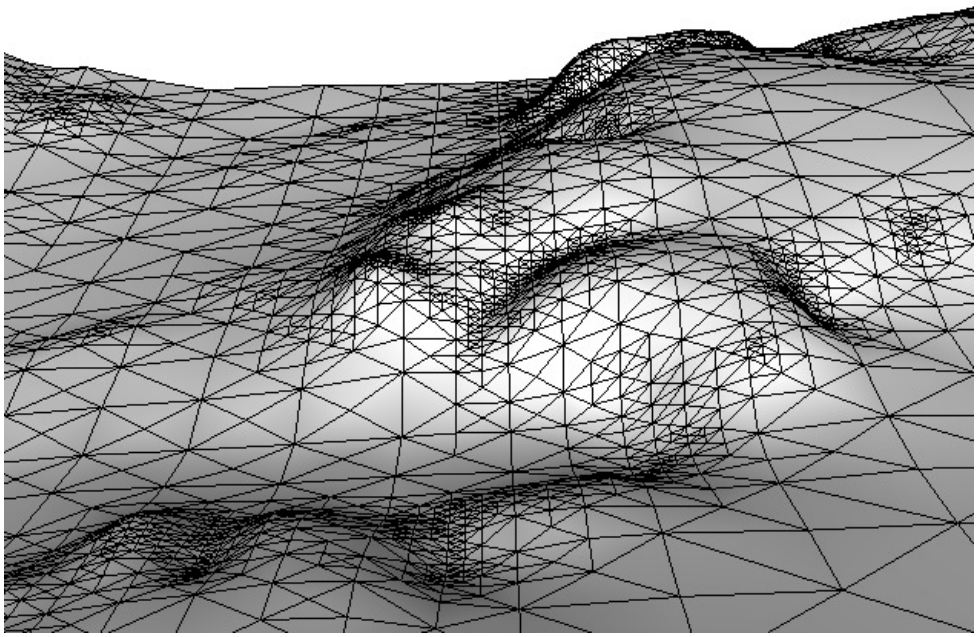


Fig. A.9: Approximation of terrain data, and triangulation imposed on the surface

coarse-to-fine scheme. With an error tolerance of 2.5 percent of $|z_{\max} - z_{\min}|$, the algorithm terminated after subdivision at level 11 and delivered the triangulation Δ_{12} with 31204 vertices. Relatively few subdivisions were done at the last two levels to capture remaining details in the terrain and meet the given tolerance. The number of vertices in Δ_{12} is only 1.5 percent of the maximum number of available vertices at that level. Algebraic multigrid (V-cycles) performed slightly better with $\lambda = 10\lambda_d$, but when λ was increased algebraic multigrid was superior. The triangulation shown in Fig. A.9 was produced by a larger tolerance to avoid too many triangles in the presentation. The mesh is finer in areas with rapidly varying topography, and thereby captures the necessary details. We also observe the natural extrapolation of the surface to areas without input data, which is due to the thin-plate energy. Algorithms with good extrapolation properties are important in many applications. For example, in geological modeling faults and horizons must be extended to intersect each other with clean cuts outside their initial domain when creating boundary-based volume models (Schneider, 2002).

A.7 Summary

The novelty of the method presented in this chapter lies in the use of binary triangulations as an effective tool for generating a nested sequence of triangulations used in a multilevel scheme for solving the scattered data approximation problem. Binary triangulations give rise to extremely simple data structures. The fact that binary triangulations are standard tools for view-dependent visualization, also makes the resulting surfaces well suited for fast rendering. In particular the method is efficient when fitting surfaces to huge scattered data sets and data unevenly distributed over the domain. When using subdivision criteria based on error measure or counting measure, the triangle density adapts automatically to the distribution of the data with nice spatial grading as can be seen in Fig. A.4 and A.9. The triangle density also reflects the variation in surface topography. Another useful feature observed by numerical experiments is the natural extrapolation of the surface to areas without data.

Even though the simple coarse-to-fine scheme works well for data without noise, convergence is significantly improved by using a standard algebraic multigrid solver when a large smoothing parameter λ is necessary for noisy data. We would probably benefit even more from a geometric multigrid solver based on the triangular grids produced by our coarse-to-fine scheme. Another interesting topic for further research is to study methods for computing a good λ when approximating data with noise.

Appendix B

Operations on Parametric Curves and Surfaces

In the fast marching method, and optionally in other upwind finite difference solvers, we use curves and surfaces on parametric form to represent the initial front. For example, a curve is given in R^2 and R^3 by the parametric representations

$$\mathbf{c}(t) = \begin{bmatrix} x(t) \\ y(t) \end{bmatrix} \quad \text{and} \quad \mathbf{c}(t) = \begin{bmatrix} x(t) \\ y(t) \\ z(t) \end{bmatrix},$$

respectively, and a surface is given in R^3 by

$$\mathbf{s}(u,v) = \begin{bmatrix} x(u,v) \\ y(u,v) \\ z(u,v) \end{bmatrix}.$$

Standard representation formats for smooth curves and surfaces are cubic Bézier curves and bicubic tensor product Bézier surfaces (Farin, 2002). Bézier surfaces are defined over regular rectangular grids and are relatively easy to construct from rectangular grid formats used by many geo-applications. They are also fast to compute and to evaluate. Bézier surfaces can also be constructed from unorganized scattered data points obtained from auto-tracking of horizons in 3D seismics, via regularization to regular grids by approximation methods such as the method in Appendix A. Both curves and surfaces on Bézier form have C^1 continuity, that is, the Cartesian coordinates x, y, z are differentiable functions with continuous first derivatives. Standard basic functions include those for evaluating the spacial position and the derivatives in a

given parameter value, from which we construct other functions. For example, the tangent vector of a curve in 2D is

$$\dot{\mathbf{c}}(t) = \frac{d\mathbf{c}(t)}{dt} = \begin{bmatrix} x'(t) \\ y'(t) \end{bmatrix},$$

and the unit normal vector in t is

$$\mathbf{n}(t) = \frac{(-y'(t), x'(t))^T}{\|\dot{\mathbf{c}}(t)\|},$$

where $\mathbf{n}(t)$ points to the left when walking in the parameter direction. Similarly, for surfaces we use the notation

$$\mathbf{s}_u(u,v) = \frac{\partial \mathbf{s}(u,v)}{\partial u}, \quad \mathbf{s}_v(u,v) = \frac{\partial \mathbf{s}(u,v)}{\partial v},$$

and the unit normal vector in (u, v) is

$$\mathbf{n}(u,v) = \frac{\mathbf{s}_u(u,v) \times \mathbf{s}_v(u,v)}{\|\mathbf{s}_u(u,v) \times \mathbf{s}_v(u,v)\|}.$$

Triangulated surfaces are also frequently used in geo-applications for representing horizons and fauls. When triangles are planar patches these surfaces do not possess the smoothness we require to obtain sufficiently smooth time of arrival fields and other derived fields. On the other hand, the triangles can be associated with cubic Bézier triangles, joined smoothly together with C^1 continuity. A practical guide to Bézier curves and surfaces is Farin (2002). This book may also serve as an excellent guide for implementaion.

B.1 Closest Point Calculation

The initialization steps of fast marching use closest point calculations to determine the distance from grid points of the finite difference grid to curves, surfaces and other geometric objects (Sect. 3.1.2). There is no closed form for distances to general parametric curves and surfaces, so we use Newton iteration to find approximations. We first demonstrate how this can be done for closest point calculation relative to a parametric curve $\mathbf{c}(t)$ by first and second order schemes.

Let $\mathbf{p}$ be a point in space (e.g. a grid point) and $\mathbf{p}_i = \mathbf{c}(t_i)$ a guess point for the closest point to $\mathbf{p}$ on the curve, see Fig. B.1. Then the following orthogonality condition determines the increment Δt of the parameter value for the next guess point $\mathbf{p}_{i+1} = \mathbf{c}(t_i + \Delta t)$ in the Newton iteraton,

$$< (\mathbf{p} - \mathbf{p}_i) - \dot{\mathbf{c}}(t_i)\Delta t \ , \ \dot{\mathbf{c}}(t_i) > = \ 0, \tag{B.1}$$

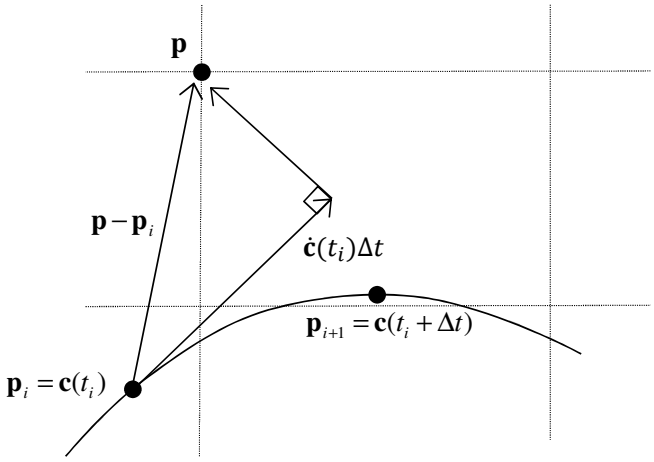


Fig. B.1: Correction step in Newton iteration for first order closest point calculation

where $< \cdot, \cdot >$ denotes the inner product. This gives the linear equation

$$\Delta t = \frac{(\mathbf{p} - \mathbf{p}_i) \cdot \dot{\mathbf{c}}(t_i)}{\dot{\mathbf{c}}(t_i) \cdot \dot{\mathbf{c}}(t_i)}.$$

In each step of the iteration, $\mathbf{p}_i$ is moved along the curve to the next guess point $\mathbf{p}_{i+1} = \mathbf{c}(t_i + \Delta t)$ until $\|\mathbf{p}_{i+1} - \mathbf{p}_i\| < \epsilon$, for some user defined geometric tolerance ϵ. Alternatively, the stop criterion can be based on the difference in parameter values $|t_{i+1} - t_i|$.

Faster convergence, and in many cases more reliable computations, can be obtained by a second order scheme where the tangent vector $\dot{\mathbf{c}}(t_i)$ in the second term of the inner product of the orthogonality condition (B.1) is replaced by the tangent vector $\dot{\mathbf{c}}(t_i + \Delta t)$ at the next guess point. This tangent vector can be approximated by the first order Taylor expansion,

$$\dot{\mathbf{c}}(t_i + \Delta t) \approx \dot{\mathbf{c}}(t_i) + \ddot{\mathbf{c}}(t_i)\Delta t,$$

where $\ddot{\mathbf{c}}(t) = d^2\mathbf{c}(t)/dt^2$. The orthogonality condition then becomes

$$< (\mathbf{p} - \mathbf{p}_i) - \dot{\mathbf{c}}(t_i)\Delta t \ , \ \dot{\mathbf{c}}(t_i) + \ddot{\mathbf{c}}(t_i)\Delta t >= 0. \tag{B.2}$$

This results in a quadratic equation in Δt,

$$-\dot{\mathbf{c}}(t_i) \cdot \ddot{\mathbf{c}}(t_i)(\Delta t)^2 + \left[(\mathbf{p} - \mathbf{p}_i) \cdot \ddot{\mathbf{c}}(t_i) - \dot{\mathbf{c}}(t_i) \cdot \dot{\mathbf{c}}(t_i)\right] \Delta t + (\mathbf{p} - \mathbf{p}_i) \cdot \dot{\mathbf{c}}(t_i) = 0.$$

The schemes can easily be extended to surfaces. Then we move the guess point for the closest point in each iteration a distance on the surface which

corresponds to increments Δu and Δv in the parameter plane. Let the guess point $\mathbf{p}_i$ have parameter value (u_i, v_i). A first order scheme is obtained by the orthogonality conditions

$$\begin{aligned} &< (\mathbf{p} - \mathbf{p}_i) - (\mathbf{s}_u(u_i, v_i)\Delta u + \mathbf{s}_v(u_i, v_i)\Delta v) \ , \ \mathbf{s}_u(u_i, v_i) > = 0 \\ &< (\mathbf{p} - \mathbf{p}_i) - (\mathbf{s}_u(u_i, v_i)\Delta u + \mathbf{s}_v(u_i, v_i)\Delta v) \ , \ \mathbf{s}_v(u_i, v_i) > = 0. \end{aligned} \tag{B.3}$$

That is, the difference vector $(\mathbf{p} - \mathbf{p}_i) - (\mathbf{s}_u(u_i, v_i)\Delta u + \mathbf{s}_v(u_i, v_i)\Delta v)$ is orthogonal to the tangent plane at the guess point $\mathbf{p}_i$. Let $\mathbf{s}_u = \mathbf{s}_u(u_i, v_i)$ and $\mathbf{s}_v = \mathbf{s}_v(u_i, v_i)$. We get a linear system with two equations in the unknowns Δu and Δv,

$$\begin{bmatrix} \mathbf{s}_u \cdot \mathbf{s}_u & \mathbf{s}_u \cdot \mathbf{s}_v \\ \mathbf{s}_u \cdot \mathbf{s}_v & \mathbf{s}_v, \mathbf{s}_v \end{bmatrix} \begin{bmatrix} \Delta u \\ \Delta v \end{bmatrix} = \begin{bmatrix} (\mathbf{p} - \mathbf{p}_i) \cdot \mathbf{s}_u \\ (\mathbf{p} - \mathbf{p}_i) \cdot \mathbf{s}_v \end{bmatrix}.$$

Similarly as for curves a second order scheme is obtained by replacing the second term of the inner products in Eq. (B.3) by $\mathbf{s}_u(u_i + \Delta u, v_i + \Delta v)$ and $\mathbf{s}_v(u_i + \Delta u, v_i + \Delta v)$, respectively, such that the difference vector is orthogonal to the tangent plane at the next guess point. A first order Taylor expansion about (u_i, v_i) gives the approximations

$$\begin{aligned} \mathbf{s}_u(u_i + \Delta u, v_i + \Delta v) &\approx \mathbf{s}_u(u_i, v_i) + \mathbf{s}_{uu}(u_i, v_i)\Delta u + \mathbf{s}_{uv}(u_i, v_i)\Delta v \\ \mathbf{s}_v(u_i + \Delta u, v_i + \Delta v) &\approx \mathbf{s}_v(u_i, v_i) + \mathbf{s}_{uv}(u_i, v_i)\Delta u + \mathbf{s}_{vv}(u_i, v_i)\Delta v. \end{aligned}$$

This results in a non-linear equation system, contrary to the linear system above,

$$\begin{aligned} \mathbf{s}_u \cdot \mathbf{s}_{uu}(\Delta u)^2 + \mathbf{s}_v \cdot \mathbf{s}_{uv}(\Delta v)^2 &+ [\mathbf{s}_u \cdot \mathbf{s}_u - (\mathbf{p} - \mathbf{p}_i) \cdot \mathbf{s}_{uu}]\Delta u \\ &+ [\mathbf{s}_v \cdot \mathbf{s}_u - (\mathbf{p} - \mathbf{p}_i) \cdot \mathbf{s}_{uv}]\Delta v \\ &+ (\mathbf{s}_v \cdot \mathbf{s}_{uu} + \mathbf{s}_u \cdot \mathbf{s}_{uv})\Delta u \Delta v = (\mathbf{p} - \mathbf{p}_i) \cdot \mathbf{s}_u \end{aligned}$$

$$\begin{aligned} \mathbf{s}_u \cdot \mathbf{s}_{uv}(\Delta u)^2 + \mathbf{s}_v \cdot \mathbf{s}_{vv}(\Delta v)^2 &+ [\mathbf{s}_u \cdot \mathbf{s}_v - (\mathbf{p} - \mathbf{p}_i) \cdot \mathbf{s}_{uv}]\Delta u \\ &+ [\mathbf{s}_v \cdot \mathbf{s}_v - (\mathbf{p} - \mathbf{p}_i) \cdot \mathbf{s}_{vv}]\Delta v \\ &+ (\mathbf{s}_v \cdot \mathbf{s}_{uv} + \mathbf{s}_u \cdot \mathbf{s}_{vv})\Delta u \Delta v = (\mathbf{p} - \mathbf{p}_i) \cdot \mathbf{s}_v \end{aligned}$$

We linearize the system such that the coefficient matrix depends on the unknowns,

$$\mathbf{A}(\mathbf{u})\mathbf{u} = \mathbf{b},$$

where $\mathbf{u} = (\Delta u, \Delta v)^T$, $\mathbf{b} = [(\mathbf{p} - \mathbf{p}_i) \cdot \mathbf{s}_u, (\mathbf{p} - \mathbf{p}_i) \cdot \mathbf{s}_v]^T$ and the elements of A are

$$
\begin{aligned}
a_{11} &= \mathbf{s}_u \cdot \mathbf{s}_{uu}\Delta u + \mathbf{s}_u \cdot \mathbf{s}_u - (\mathbf{p} - \mathbf{p}_i) \cdot \mathbf{s}_{uu} + (\mathbf{s}_v \cdot \mathbf{s}_{uu} + \mathbf{s}_u \cdot \mathbf{s}_{uv})\Delta v, \\
a_{12} &= \mathbf{s}_v \cdot \mathbf{s}_{uv}\Delta v + \mathbf{s}_u \cdot \mathbf{s}_v - (\mathbf{p} - \mathbf{p}_i) \cdot \mathbf{s}_{uv}, \\
a_{21} &= \mathbf{s}_u \cdot \mathbf{s}_{uv}\Delta u + \mathbf{s}_u \cdot \mathbf{s}_v - (\mathbf{p} - \mathbf{p}_i) \cdot \mathbf{s}_{uv}, \\
a_{22} &= \mathbf{s}_v \cdot \mathbf{s}_{vv}\Delta v + \mathbf{s}_v \cdot \mathbf{s}_v - (\mathbf{p} - \mathbf{p}_i) \cdot \mathbf{s}_{vv} + (\mathbf{s}_v \cdot \mathbf{s}_{uv} + \mathbf{s}_u \cdot \mathbf{s}_{vv})\Delta u.
\end{aligned}
$$

A simple iteration scheme based on successive substitutions solves the system in a few iterations,

$$\mathbf{A}(\mathbf{u}^k)\mathbf{u}^{k+1} = \mathbf{b}, \qquad k = 0, 1, \ldots$$

with start vector $\mathbf{u}^0 = \mathbf{0}$.

A good initial guess for $\mathbf{p}_i$ is necessary for convergence to the correct solution for all the iterative schemes above. In the context of fast marching the point $\mathbf{p}$ will be a grid point in a small neighborhood of a geometric object. These points are collected in the initialization step by marching along the curve or the surface object and then given a reference to a parameter value of a nearby point on the geometry. This serves as a sufficiently good guess for the closest point if the computational grid is sufficiently dense.

B.2 Intersection with Straight Lines

Another basic operation used in the initialization step of fast marching in Sect. 3.1.2 is intersection of a straight line with a curve or a surface. A Newton iteration scheme similar to the closest point calculation does the job. Let $\vec{\mathbf{d}}$ be the direction of a line l that passes through a (grid) point $\mathbf{p}$, and let $\mathbf{p}_i = \mathbf{c}(t_i)$ be an initial guess of the intersection point between l and the curve $\mathbf{c}(t)$. A first order scheme is obtained by replacing the tangent vector $\dot{\mathbf{c}}(t_i)$ in the second term of the inner product in Eq. (B.1) with a fixed direction $\perp \vec{\mathbf{d}}$ orthogonal to l,

$$< (\mathbf{p} - \mathbf{p}_i) - \dot{\mathbf{c}}(t_i)\Delta t \ , \ \perp \vec{\mathbf{d}} >= 0.$$

This gives the parameter correction

$$\Delta t = \frac{(\mathbf{p} - \mathbf{p}_i) \cdot \perp \vec{\mathbf{d}}}{\dot{\mathbf{c}}(t_i) \cdot \perp \vec{\mathbf{d}}}$$

for each step in the Newton iteration.

A similar approach can be used for intersecting a straight line l with a surface. Let l have direction $\vec{\mathbf{d}}$ and be passing through a (grid) point $\mathbf{p}$, and let $\perp \vec{\mathbf{d}}_1$ and $\perp \vec{\mathbf{d}}_2$ be two non-collinear vectors that both are orthogonal to $\vec{\mathbf{d}}$. We modify the orthogonality condition for closest point calculation above to

$$< (\mathbf{p} - \mathbf{p}_i) - (\mathbf{s}_u(u_i, v_i)\Delta u + \mathbf{s}_v(u_i, v_i)\Delta v) \ , \ \perp \vec{\mathbf{d}}_1 > = 0$$
$$< (\mathbf{p} - \mathbf{p}_i) - (\mathbf{s}_u(u_i, v_i)\Delta u + \mathbf{s}_v(u_i, v_i)\Delta v) \ , \ \perp \vec{\mathbf{d}}_2 > = 0.$$

That is, the difference vector in the Newton iteration is now orthogonal to a plane spanned by $\perp \vec{\mathbf{d}}_1$ and $\perp \vec{\mathbf{d}}_2$.

Appendix C

Finite Difference Operators for Derivatives and Curvature

The upwind schemes require first and second order backward and forward differencing for computing approximations to the first and second derivatives. Second derivatives are required when calculating curvature. This section contains a catalogue of finite difference operators on Cartesian grids with equally spaced grid lines in each space direction. Operators for directional derivatives used in the dynamic upwind stencil construction for anisotropic problems (Sect. 3.1.5), can be written down accordingly by reinterpreting the indices and the spacing Δx.

C.1 Numerical Derivatives

Second order backward approximation to the first derivative

$$T_x \approx \frac{3T_i - 4T_{i-1} + T_{i-2}}{2\Delta x} = D^{-x}T + \frac{\Delta x}{2} D^{-x-x}T,$$

where

$$D^{-x}T = \frac{T_i - T_{i-1}}{\Delta x}, \text{ and } D^{-x-x}T = \frac{T_i - 2T_{i-1} + T_{i-2}}{\Delta x^2}.$$

Second order forward approximation to the first derivative

$$T_x \approx -\frac{3T_i - 4T_{i+1} + T_{i+2}}{2\Delta x} = D^{+x}T - \frac{\Delta x}{2} D^{+x+x}T,$$

where

$$D^{+x+x}T = \frac{T_{i+2} - 2T_{i+1} + T_i}{\Delta x^2}.$$

Second order central approximation to the first derivative

$$T_x \approx \frac{T_{i-2} - 8T_{i-1} + 8T_{i+1} - T_{i+2}}{12\Delta x}.$$

First order central approximation to the first derivative

$$T_x \approx \frac{T_{i+1} - T_{i-1}}{2\Delta x}.$$

Second order backward approximation to the second derivative

$$T_{xx} \approx \frac{2T_i - 5T_{i-1} + 4T_{i-2} - T_{i-3}}{\Delta x^2}.$$

Another formula derived by differentiating the backward version of T_x above is

$$T_{xx} \approx \frac{3T_i - 7T_{i-1} + 5T_{i-2} - T_{i-3}}{2\Delta x^2}.$$

Second order central approximation to the second derivative

$$T_{xx} \approx \frac{-T_{i-2} + 16T_{i-1} - 30T_i + 16T_{i+1} - T_{i+2}}{12\Delta x^2}.$$

Second order central approximation to the mixed derivative

$$T_{xy} \approx \frac{T_{i+1,j+1} - T_{i-1,j+1} - T_{i+1,j-1} + T_{i-1,j-1}}{4\Delta x \Delta y}.$$

C.2 Curvature as Divergence of the Unit Normal

The scalar curvature in 2D can be expressed as the divergence of the unit normal vector,

$$\kappa = \nabla \cdot \mathbf{n} = \nabla \cdot \frac{\nabla T}{\|\nabla T\|} = \frac{\partial}{\partial x} \frac{T_x}{(T_x^2 + T_y^2)^{1/2}} + \frac{\partial}{\partial y} \frac{T_y}{(T_x^2 + T_y^2)^{1/2}}.$$

Applying the chain rule, the first term on the right hand side becomes

$$\frac{\partial}{\partial x}\,\frac{T_x}{(T_x^2+T_y^2)^{1/2}} =$$

$$\frac{T_{xx}(T_x^2+T_y^2)^{1/2} - T_x\frac{1}{2}(T_x^2+T_y^2)^{-1/2}(2T_xT_{xx}+2T_yT_{xy})}{(T_x^2+T_y^2)} =$$

$$\frac{T_{xx}(T_x^2+T_y^2) - T_x(T_xT_{xx}+T_yT_{xy})}{(T_x^2+T_y^2)^{3/2}} = \frac{T_{xx}T_y^2 - T_xT_yT_{xy}}{(T_x^2+T_y^2)^{3/2}},$$

and the second term on the right hand side becomes

$$\frac{\partial}{\partial y}\,\frac{T_y}{(T_x^2+T_y^2)^{1/2}} = \frac{T_{yy}T_x^2 - T_xT_yT_{xy}}{(T_x^2+T_y^2)^{3/2}}.$$

The curvature can then be written

$$\kappa = \frac{T_{xx}T_y^2 - 2\,T_xT_yT_{xy} + T_{yy}T_x^2}{(T_x^2+T_y^2)^{3/2}},$$

where the numerical derivatives can be taken from Sect. C.1. In three dimensions the Gaussian curvature and the mean curvature can be calculated similarly from first and second derivatives (Goldman, 2005).

Bibliography

Arge, E., Dæhlen, M., and Tveito, A. (1995). Approximation of scattered data using smooth grid functions. *Journal of Computational and Applied Mathematics*, **59**, 191–205.

Bobillo-Ares, N., Aller, J., Bastida, F., Lisle, R., and Toimil, N. (2006). The problem of area change in tangential longitudinal strain folding. *Journal of Structural Geology*, **28**(10), 1835–1848.

Braccini, E., de Boer, W., Hurst, A., Huuse, M., Vigorito, M., and Templeton, G. (2008). Sand injectites. *Oil Field Review (Summer)*, pages 34–49.

Brandt, A. (1986). Algebraic multigrid theory: The symmetric case. *Applied Mathematics and Computation*, **19**, 23–56.

Brandt, A., McCormick, S. F., and Ruge, J. (1984). Algebraic multigrid (AMG) for sparse matrix equations. In D. J. Evans, editor, *Sparsity and its Applications*, pages 257–284. Cambridge University Press.

Briggs, W. L., Henson, V. E., and McCormick, S. F. (2000). *A multigrid tutorial: second edition*. Society for Industrial and Applied Mathematics.

Crandall, M. G. and Lions, P. L. (1983). Viscosity solutions of Hamilton-Jacobi equations. *Transactions of the American Mathematical Society*, **277**(1), 1–43.

Crandall, M. G., Evans, L. C., and Lions, P. L. (1984). Some properties of viscosity solutions of Hamilton-Jacobi equations. *Transactions of the American Mathematical Society*, **282**(2), 487–502.

Do Carmo, M. (1976). *Differential Geometry of Curves and Surfaces*. Prentice Hall, Englewood Cliffs, NJ.

Durand-Riard, P., Caumon, G., and Muron, P. (2010). Balanced restoration of geological volumes with relaxed meshing constraints. *Computers & Geosciences*, **36**(4), 441–452.

Dyn, N., Levin, D., and Rippa, S. (1990). Data dependent triangulations for piecewice linear interpolation. *IMA Journal of Numerical Analysis*, **10**, 137–154.

Farin, G. (2002). *Curves and Surfaces for CAGD: A Practical Guide*. Academic Press, 5th edition.

Floater, M. (2003). Mean value coordinates. *Computer Aided Geometric Design*, **20**(1), 19–27.

Floater, M. S. (1998). How to approximate scattered data by least squares. Technical Report STF42 A98013, SINTEF, Oslo.

Forsey, D. R. and Bartels, R. H. (1988). Hierarchical B-spline refinement. In *ACM Computer Graphics (SIGGRAPH '88 Proceedings)*, pages 205–212.

Forsey, D. R. and Bartels, R. H. (1995). Surface fitting with hierarchical splines. *ACM Transactions on Graphics*, **14**, 134–161.

Franke, R. (1982). Scattered data interpolation: Tests of some methods. *Math. Comp.*, **38**, 181–200.

Gilardet, M., Guillon, S., Jobard, B., and Komatitsch, D. (2013). Seismic image restoration using nonlinear least squares shape optimization. In *Procedia Computer Science*, volume 18, pages 732–741. Elsevier B.V.

Gillberg, T. (2011). A semi-ordered fast iterative method (SOFI) for monotone front propagation in simulations of geological folding. In F. Chan, D. Marinova, and B. Anderssen, editors, *MODSIM2011, 19th International Congress on Modelling and Simulation*, pages 631–647. Modelling and Simulation Society of Australia and New Zealand.

Gillberg, T. (2013). *Fast and accurate front propagation for simulation of geological folds*. Ph.D. thesis, University of Oslo.

Gillberg, T., Hjelle, Ø., and Bruaset, A. M. (2012). Accuracy and efficiency of stencils for the eikonal equation in earth modelling. *Computational Geosciences*, **16**(4), 933–952.

Gillberg, T., Bruaset, A. M., Hjelle, Ø., and Sourouri, M. (2014). Parallel two-scale methods for solution of static Hamilton-Jacobi equations. Submitted.

Goldman, R. (2005). Curvature formulas for implicit curves and surfaces. *Computer Aided Geometric Design*, **22**(7), 632–658.

Golub, G. H. and Loan, C. F. (1996). *Matrix Computations*. John Hopkins University Press, Baltimore and London, third edition.

Grasemann, B., Martel, S., and Passchier, C. (2005). Reverse and normal drag along a fault. *Journal of Structural Geology*, **27**(6), 999–1010.

Greiner, G. and Hormann, K. (1997). Interpolating and approximating scattered 3D data with hierarchical tensor product B-splines. In A. Méhauté, C. Rabut, and L. L. Schumaker, editors, *Surface Fitting and Multiresolution Methods*, pages 163–172.

Guskov, I. (1998). Multivariate subdivision schemes and divided differences. Technical report, Department of Mathematics, Princeton University.

Guskov, I., Sweldens, W., and Schröder, P. (1999). Multiresolution signal processing for meshes. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 325–334. ACM Press/Addison-Wesley Publishing Co.

Hardy, S. and McClay, K. (1999). Kinematic modelling of extensional fault-propagation folding. *Journal of Structural Geology*, **21**(7), 695–702.

Harlow, F. and Welch, J. (1965). Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. *Physics of Fluids*, **8**(12), 2182–2189.

Hjelle, Ø. (2001). Approximation of scattered data with multilevel B-splines. Technical Report STF42 A01011, SINTEF Applied Mathematics.

Hjelle, Ø. and Dæhlen, M. (2005). Multilevel least squares approximation of scattered data over binary triangulations. *Computing and Visualization in Science*, **8**(2), 83–91.

Hjelle, Ø. and Dæhlen, M. (2006). *Triangulations and Applications*. Springer Verlag.

Hjelle, Ø., Petersen, S. A., and Bruaset, A. M. (2013). A numerical framework for modeling folds in structural geology. *Mathematical Geosciences*, **45**(3), 255–276.

Hjelle, Ø. and Petersen, S. A. (2011). A Hamilton-Jacobi framework for modeling folds in structural geology. *Mathematical Geosciences*, **43**(7), 741–761.

Hu, J., Tong, C., and Tuminaro, R. S. (2000). ML 2.0 Smoothed aggregation user's guide. Technical Report SAND2001-8028, Sandia National Laboratories, Albuquerque NM.

Hudleston, P.-J. (1973). Fold morphology and some geometrical implications of theories of fold development. *Tectonophysics*, **16**, 1–46.

Jeong, W.-K. and Whitaker, R. T. (2008). A fast iterative method for eikonal equations. *SIAM Journal on Scientific Computing*, **30**(5), 2512–2534.

Jeong, W.-K., Fletcher, P. T., Tao, R., and Whitaker, R. (2007). Interactive visualization of volumetric white matter connectivity in DT-MRI using a parallel-hardware Hamilton-Jacobi solver. *IEEE-Transactions-on-Visualization-and-Computer-Graphics*, **13**(6), 1480–1487.

Jin, G. and Groshong Jr., R. H. (2006). Trishear kinematic modeling of extensional fault-propagation folding. *Journal of Structural Geology*, **28**(1), 170–183.

Kao, C.-Y., Osher, S., and Tsai, Y.-H. (2005). Fast sweeping methods for static Hamilton-Jacobi equations. *SIAM Journal on Numerical Analysis*, **42**(6), 2612–2632.

Kornhauser, E. (1953). Ray theory for moving fluids. *Journal of the Acoustical Society of America*, **25**(5), 945–949.

Kreyszig, E. (1991). *Differential Geometry*. Dover Publications, New York.

Laurent, G., Caumon, G., Bouziat, A., and Jessell, M. (2013). A parametric method to model 3D displacements around faults with volumetric vector fields. *Tectonophysics*, **590**(0), 83–93.

Lee, S., Wolberg, G., and Shin, S. Y. (1997). Scattered data interpolation with multilevel B-splines. *IEEE-Transactions-on-Visualization-and-Computer-Graphics*, **3**(3), 229–244.

LeVeque, R. J. (1992). *Numerical Methods for Conservation Laws*. Lectures in mathematics ETH Zürich. Birkhäuser, Basel.

Lin, Q. (2003). *Enhancement, Extraction, and Visualization of 3D Volume Data*. Ph.D. thesis, Linköping University.

Lindstrom, P. and Pascucci, V. (2002). Terrain simplification simplified: A general framework for view-dependent out-of-core visualization. *IEEE Transactions on Visualization and Computer Graphics*, **8**(3), 239–254.

Lindstrom, P., Koller, D., Ribarsky, W., Hodges, L. F., Faust, N., and Turner, G. A. (1996). Real-time, continuous level of detail rendering of height fields. In *ACM SIGGRAPH 96*, pages 109–118.

Maerten, L. and Maerten, F. (2006). Chronologic modeling of faulted and fractured reservoirs using geomechanically based restoration: Technique and industry applications. *AAPG Bulletin*, **90**(8), 1201–1226.

Mallet, J.-L. (2004). Space-time mathematical framework for sedimentary geology. *Mathematical Geology*, **36**(1), 1–32.

Mallet, J.-L. (2008). *Numerical Earth Models*. EAGE publications.

Moretti, I., Lepage, F., and Guiton, M. (2006). KINE3D : a new 3D restoration method based on a mixed approach linking geometry and geomechanics. *Oil & Gas Science And Technology*, **61**(2), 277–289.

Nishida, T., Sugihara, K., and Kimura, M. (2007). Stable marker-particle method for the Voronoi diagram in a flow field. *Journal of Computational and Applied Mathematics*, **202**(2), 377–391.

Pajarola, R. B. (1998). Large scale terrain visualization using the restricted quadtree triangulation. In *Proceedings of the conference on Visualization '98*, pages 19–26. IEEE Computer Society Press.

Petersen, S. A. and Hjelle, Ø. (2008). Earth recursion, an important component in sheared earth model builders. In *EAGE 70th Conference & Exhibition, Extended Abstracts*.

Petersen, S. A. and Skjei, N. (2005). Geological modeling of intrusive bodies. In *EAGE 67th Conference & Exhibition, Extended Abstracts*.

Petersen, S. A., Hjelle, Ø., and Jensen, S. L. (2007). Earth modelling using distance fields derived by Fast Marching. In *EAGE 69th Conference & Exhibition, Extended Abstracts*.

Petersen, S. A., Hjelle, Ø., Hustoft, S., Haubiers, M., and Rasmussen, R. (2012). Process based data-restoration and model-reconstruction workflow for seismic interpretation and model building. In *EAGE 74th Conference & Exhibition, Extended Abstracts*.

Pons, J.-P. and Boissonnat, J.-D. (2007). A Lagrangian approach to dynamic interfaces through kinetic triangulation of the ambient space. *Computer Graphics Forum*, **26**(2), 227–239.

Powell, M. J. D. (1992). The theory of radial basis function approximation in 1990. In W. Light, editor, *Advances in Numerical Analysis, Vol II*, pages 105–210. Oxford Science Publications.

Qian, J., Zhang, Y.-T., and Zhao, H.-K. (2007). A fast sweeping method for static convex Hamilton-Jacobi equations. *Journal of Scientific Computing*, **31**(1/2), 237–271.

Ragan, D. M. (2009). *Structural Geology. An Introduction to Geometrical Techniques.* Cambridge University Press, 4th edition.

Ramsay, J. G. (1967). *Folding and Fracturing of Rocks*. McGraw-Hill, New York and London.

Ramsay, J. G. and Huber, M. I. (1987). *The Techniques of Modern Structural Geology: Folds and Fractures*. Academic Press.

Rippa, S. (1992). Adaptive approximation by piecewise linear polynomials on triangulations of subsets of scattered data. *SIAM Journal on Scientific and Statistical Computing*, **18**(3), 1123–1141.

Röttger, S., Heidrich, W., Slusallek, P., and Seidel, H.-P. (1998). Real-time generation of continuous levels of detail for height fields. In V. Skala, editor, *Proceedings of 1998 International Conference in Central Europe on Computer Graphics and Visualization*, pages 315–322.

Santi, M. R. and Martha, L. F. (2003). A finite element approach for geological section reconstruction. In *XXIV Iberian Latin-American Congress on Computational Methods in Engineering*, pages 1–12.

Schneider, S. (2002). *Pilotage Automatique de la Construction de Modèles Géologiques Surfaciques*. Ph.D. thesis, Université Jean Monnet et Ecole Nationale Supérieure des Mines de Saint-Etienne.

Sedgewick, R. and Wayne, K. (2010). *Algorithms, 4th edition*. Addison-Wesley.

Sethian, J. A. (1985). Curvature and the evolution of fronts. *Communications in Mathematical Physics*, **101**, 487–499.

Sethian, J. A. (1996). A fast marching level set method for monotonically advancing fronts. *Proceedings of the National Academy of Sciences*, **93**, 1591–1595.

Sethian, J. A. (1999a). Curvature flow and entropy conditions applied to grid generation. *J. Comp. Phys*, **115**, 440–454.

Sethian, J. A. (1999b). *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science*. Cambridge University Press.

Sethian, J. A. and Vladimirsky, A. (2003). Ordered upwind methods for static Hamilton-Jacobi equations: Theory and algorithms. *SIAM Journal on Numerical Analysis*, **41**(1), 325–363.

Shepard, D. (1968). A two-dimensional interpolation function for irregularly spaced data. In *Proc. 23rd Nat. Conf. ACM*, pages 517–524.

Srivastava, D. C. and Lisle, R. J. (2004). Rapid analysis of fold shape using Bézier curves. *Journal of Structural Geology*, **26**(9), 1553–1559.

Srivastava, D. C. and Shah, J. (2008). The "isogon rosette" method for rapid estimation of strain in flattened folds. *Journal of Structural Geology*, **30**(4), 444–450.

Stabler, C. L. (1968). Simplified Fourier analysis of fold shapes. *Tectonophysics*, **6**(4), 343–350.

Toxopeus, G., Thorbecke, J., Wapenaar, K., Petersen, S., Slob, E., and Fokkema, J. (2008). Simulating migrated and inverted seismic data by filtering a geologic model. *Geophysics*, **73**(2).

Twiss, R. J. and Moores, E. M. (2007). *Structural Geology*. W. H. Freeman, New York, 2nd edition.

Underwood, P. (1983). Dynamic relaxation (in structural transient analysis). *Computational methods for transient analysis*, pages 245–265.

Velho, L. and Gomes, J. (2000). Variable resolution $4 - k$ meshes: Concepts and applications. *Computer Graphics Forum*, **19**(4), 195–212.

Vladimirsky, A. (2001). *Fast Methods for Static Hamilton-Jacobi Partial Differential Equations*. Ph.D. thesis, Department of Mathematics, University of California, Berkeley.

Von Golitschek, M. and Schumaker, L. L. (1990). Data fitting by penalized least squares. In J. C. Mason and M. G. Cox, editors, *Algorithms for Approximation II*, pages 210–227. Chapman & Hall.

Weng, Y., Xu, W., Wu, Y., Zhou, K., and Guo, B. (2006). 2D shape deformation using nonlinear least squares optimization. *Visual Computer*, **22**(9), 653–660.

Zhao, H. (2005). A fast sweeping method for eikonal equations. *Mathematics of Computations*, **74**(250), 603–627.

Curriculum Vitae

Øyvind Hjelle got his M.Sc. in 1980 from the Norwegian Institute of Technology (NTH) in Trondheim[1]. After four years with technical consultancy (Stømme AS), he joined SINTEF Applied Mathematics in Oslo, first as a research scientist and technical project manager, and later partly as research director. The technical work at SINTEF was in diverse areas of scientific programming, including geometric modeling, medical imaging, partial differential equations, neuroinformatics etc. Collaborators and customers were in the automotive and and aerospace industry (including a stay at Boeing in Seattle for three months), the surveying and mapping industry, geological modeling for the oil industry, and more.

Hjelle was associate professor at Department of Informatics, University of Oslo (UiO), from 2000 to 2010 (education, supervision and research in 20% and 40% positions). In 2002 he joined Simula Research Laboratory as a visiting research scientist, and later as a full time researcher. From 2005 he has been employed by Kalkulo AS, a subsidiary of Simula, and is currently chief scientific programmer. Work at Kalkulo includes scientific programming, and research and publication in collaboration with the oil industry in the realm of structural geology and geophysics. He is also involved in supervision of PhD students in collaboration with Simula's Computational Geosciences group. He is the author of the book "Triangulations and Applications", Springer Verlag (Hjelle and Dæhlen, 2006), that has been used at graduate level at Department of Informatics at UiO.

The main collaborator from the industry while employed by Simula and Kalkulo AS has been Statoil ASA, one the world's largest crude oil and gas suppliers. Both software development and research have been conducted, and this collaboration have also resulted in various publications that the major part of this thesis is based on. The research has also led to a fruitful collaboration with Delft University of Technology, the host of Hjelle's PhD.

[1]NTH is now part of the Norwegian University of Science and Technology (NTNU).