Fast and accurate front propagation for simulation of geological folds

by
Tor Gillberg

Thesis submitted for the degree of Philosophiae Doctor
Department of Informatics
Faculty of Mathematics and Natural Sciences
University of Oslo
October 2013

Preface

Early 2010 I received an industrial PhD scholarship with Kalkulo AS, a commercial subsidiary of the Simula Research Laboratory. During my time with Kalkulo, I have been part of the Compound Earth Simulator development team. Compound is Statoil technology in which models of the inner earth are transformed between different structural states. These transformations are used repeatedly, and must be performed fast in order for the software to be user friendly. This thesis summarises my research on numerical aspects of the Compound Earth Simulator.

I would like to thank my supervisors Are Magnus Bruaset, Christian Tarrou, Aslak Tveito and my informal but acting supervisor Øyvind Hjelle. Throughout my time at Simula, Are Magnus Bruaset has helped steer my research towards useful and interesting subjects. His scientific understanding and help has been invaluable. As part of Kalkulo, I have participated in several interesting meetings with key personnel in Statoil including the visionary Compound technology leader Steen A. Petersen. The mathematical model and numerical framework of Compound was developed by Øyvind Hjelle at Kalkulo. Hjelle's work has been the base of my research, and his detailed understanding and interest has been useful and encouraging.

Simula provides a friendly and welcoming environment for many talented researchers and good friends. Answers to many questions have often been found in the office next door, and I am thankful for the help from Mohammed Sourouri, Tangui Morvan, Omar al-Khayat and several other colleagues at Simula. My family have been supportive throughout my work, thank you all for being so understanding. Finally, none of my research would ever have been completed without the understanding and support from Helena, thank you.

Contents

List of Papers

- **Paper I**

  Accuracy and efficiency of stencils for the eikonal equation in earth modelling
  Tor Gillberg, Øyvind Hjelle, Are Magnus Bruaset
  Published in *Computational Geosciences 16*, number 4 (2012), pages 933-952.

- **Paper II**

  A semi-ordered fast iterative method (SOFI) for monotone front propagation in simulations of geological folding
  Tor Gillberg
  Published in the proceedings of *MODSIM 2011, 19th International Congress on Modelling and Simulation*, pages 641-647.

- **Paper III**

  A new parallel 3D front propagation algorithm for fast simulation of geological folds
  Tor Gillberg, Mohammed Sourouri, Xing Cai
  Published in *Procedia Computer Science, 9*, pages 947-955. Proceedings of the International Conference on Computational Science, ICCS 2012.

- **Paper IV**

  Parallel solvers for static Hamilton-Jacobi equations in three dimensions
  Tor Gillberg, Are Magnus Bruaset, Mohammed Sourouri, Øyvind Hjelle
  Simula Technical Report, July 2013. Under preparation for journal publication.

Introduction

At the turn of the twentieth century, water-well drillers in Texas accidentally discovered several oil fields. The value of oil and gas was soon understood, and wells were drilled at more or less random locations in search for more. Eventually, discoveries of new reservoirs became less frequent and more scientific approaches were introduced. The field of exploration geology has since been invaluable to the energy-demanding industry and society.

Both seismic imaging and geophysical simulations pose challenging computational problems in which large amounts of data are processed. Since the early days of machine-based computing, the petroleum industry has invested enormous amounts of money in computational resources and helped push the high performance computing field forward. Several of the largest supercomputers during the last 20 years were in fact developed with geophysical applications in mind [1].

Oil and gas is created as buried organic matter is turned into hydrocarbons by geophysical processes acting over long periods of time. Layers of rocks do not behave like solids when observed over millions of years, and simple layered geologies are slowly transformed into complex structural formations. Reservoirs are formed as hydrocarbons accumulate in traps formed by geological structures. However, oil that once was in one location may migrate to other places since the geology changes over time.

Several new petroleum reservoirs have been found during the last decades. One example is the deep-water resources, for example the Gulf of Mexico, where the cost of drilling one well is more than 100 million US dollars. In order to minimize the risk of drilling expensive dry wells we should try to understand how, and in which sequence, geological formations were created. A model of the geological

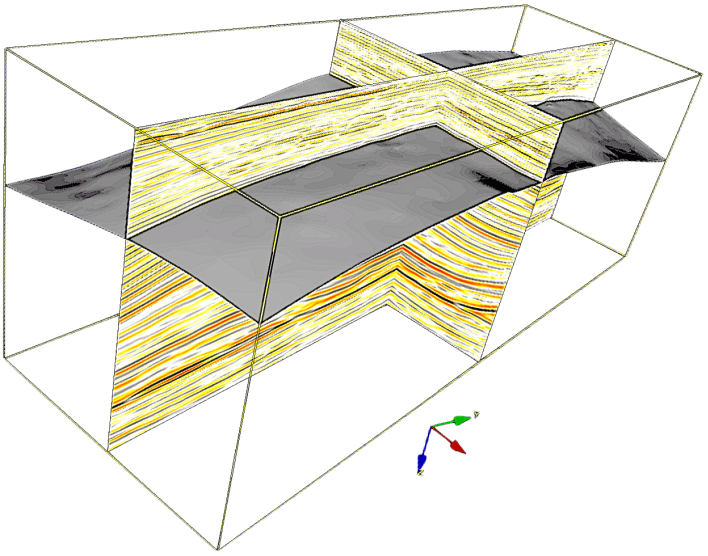
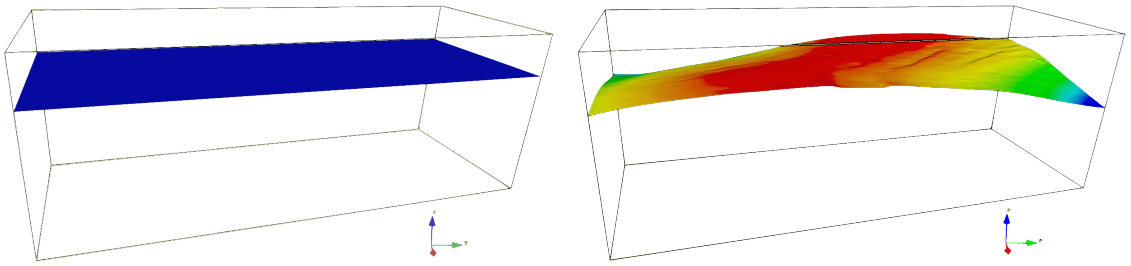


Figure 1: A geological horizon and two seismic cross sections in a three-dimensional offshore seismic volume, shown in Statoil's Compound Earth Simulator.



(a) (b)

Figure 2: (a) A flat geological horizon. (b) The same horizon several millions of years later when it has been deformed by physical processes and buried under assembled sediments.

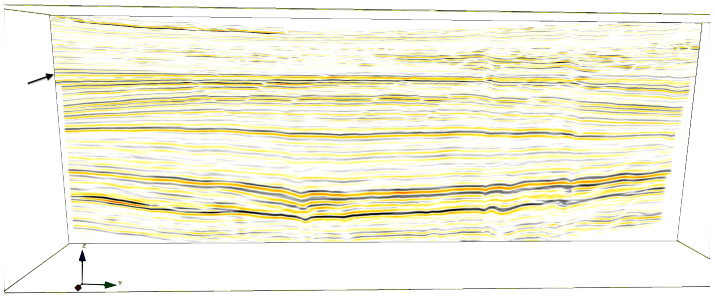
evolution helps us understand where the oil and gas might be located today. Such information can also be used to optimize the recovery phase, during which hydrocarbons are recovered from known reservoirs. Several different software tools are used for the purpose of petroleum exploration and recovery. The *Compound Earth Simulator* (CES) is one such exploration software being developed by Kalkulo for Statoil, based on modelling concepts created by the Statoil researcher Steen A. Petersen [39]. The interface separating two layers of rocks is referred to as a geological horizon, and is a geological layer with no thickness. Figure 1 shows a modelled horizon and two seismic cross sections in the CES software. The three-dimensional seismic data is part of a larger offshore seismic volume.

1 The Compound Earth Simulator

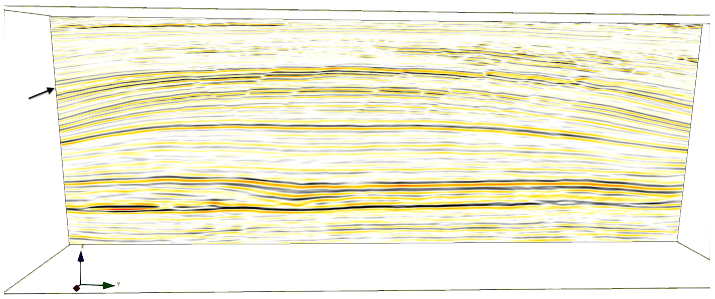
Geological structures are formed by a complicated interplay between different processes, acting on time scales ranging from seconds to millions of years [10]. Slowly moving tectonic plates cause layers to bend, fold, or break into faults. Far beneath the surface, chemical reactions, high pressure and temperature transform and create rocks in processes known as metamorphism and diagenesis. At the same time, sediments are slowly eroded and assembled to form new layers on top of the older ones. Different physical processes transform the geology into complex structural formations through a series of distinct events. Structural restoration is a methodology to validate geological models by reversing the effects of these transformations [14, 57]. In a restoration step, the model is mapped back in time to an often less complicated geological state. After restoring one deformation, earlier deformations are identified and a new restoration is initiated.

Over several years, Statoil and Kalkulo have developed a novel paradigm for highly interactive modelling of complicated geological scenarios. This methodology is implemented in CES, and has proven to be a powerful tool for geological modelling in connection to hydrocarbon exploration and production [2, 5, 39, 41]. Through a careful separation of complicated geological processes to simpler sub-processes, the present-day geology is described as the realisation of a series of geological events along a timeline [5, 40], similar to the workflow of structural restoration [41]. Models of the geology can be transformed between different structural states along the timeline. Not only the structures are restored, but also rock properties are restored in every step of the CES workflow.

Consider the “young” planar horizon in figure 2(a). During millions of years, this horizon is buried under new layers of sediments, and transformed to the shape in figure 2(b) (same as in figure 1). This transformation was reversed in CES, so that the bent horizon, 2(b), was transformed to it’s younger flat state, 2(a). In fact, the entire three-dimensional data set was transformed to match the flat horizon. Figure 3(a) shows a cross section from the restored seismic, and 3(b) the same cross section in the original seismic. The horizon being flattened is marked with an arrow in both figures.



(a)



(b)

Figure 3: Vertical cross sections in a three-dimensional offshore seismic data volume. (a) A cross section in the restored seismic dataset, where the arrow points at the flattened horizon shown in figure 2(a). (b) The same cross section in the original seismic volume, where the bent horizon in figures 2(b) and 1 has been marked with an arrow.

By simulating the geological history both backward and forward in time, several possible scenarios can be investigated. Examination of an ensemble of possible models gives an understanding of the uncertainty in the modelling. It also provides insights in the possible ways that the geology could have been formed. The engineers and geologists using the CES system can populate the restored model with properties such as porosity, permeability and chemical composition. Since the restored model is a simplified version of the complex unrestored model, this population of properties can be done with greater confidence than if it was performed in the unrestored model. In the forward simulation, all properties are transformed to their location in the complex unrestored model of the present day geology. By repeating this process, the understanding of the reservoir formation can be greatly enhanced, allowing engineers and geologists to optimise the exploration and recovery phases in the search for more oil and gas.

Modelling software. The CES software is the only modelling software that is considered in this thesis. However, there exist several alternative tools based on other methodologies. For instance, Midland Valley develop a commercial structural restoration software tool called 2DMove and 3DMove. Compared to CES, this software is more of a model building framework than a restoration tool. Paradigm has structural restoration tools in their GOCAD geomodeller, that have been upgraded to the SKUA software. SKUA's structural restoration is based on a (u,v,t) transformation [35] of the geological structures. Structures are parameterised by the spatial parameters u and v, and mapped between geological times t. The Kine3D tool combines the GOCAD mappings with a finite element code in order to investigate geomechanical properties such as stress and strain [36].

Structural transformations can be designed to minimize the difference in certain properties of meshes

describing geological structures. Such mesh properties often includes the mean value coordinates [15] or the edge lengths [17] of the elements in the mesh. For instance, the transformations in the earth model update procedure described in [55] are based on the mean value coordinates. A finite element based fault restoration procedure in two dimensions have been implemented in Recon [11], and extended to three dimensions as a plug-in to GOCAD [12].

All mentioned restoration approaches first mesh geological structures and then enforce structural changes on the meshed geometry. In contrast, the CES methodology does not need to describe structures with meshes since geological structures are modelled by implicit surfaces. Transformation of structures are made on basis of a "distance" to the implicit surface. The distances are not always measured in a Euclidean sense, but instead computed as the arrival time of a front, propagating from the implicit surface and to the rest of the domain. The arrival times, or distances, can thus be seen as three-dimensional distance field. Observe that geological structures can be mapped between very general shapes by combining two distance fields. In the remainder of this thesis, only restorations using CES are considered.

2 Motivation of research

In the CES workflow, geological models are repeatedly restored and transformed between different structural states [40, 41]. For the software to be user friendly, the distance fields used in the transformations must be computed rapidly while still being accurate. This requirement poses a computational challenge since the governing differential equation is nonlinear, and computational grids must be finely spaced to achieve high numerical accuracy. Computations associated with updating a node in the grid are expensive in terms of both floating-point operations and logical branching.

Goal. This thesis investigates efficient methods for accurate simulations of propagating fronts, as needed for the computation of distance fields in the CES software. The goal is to design and test methods that are able to solve the computational challenges faster than the current CES implementation by taking advantage of state-of-the-art hardware platforms for numerical computations.

Main results. With the novel algorithms presented in this thesis, the time needed to simulate a folded structure is reduced up to 99.5%, thereby allowing interactive modelling on basis of large three-dimensional grids. The new algorithms are currently being transferred to the CES software, thus bridging research and technology development in a way that supports the goals of the industrial PhD scheme, as defined by the Research Council of Norway.

3 Front propagation

Many physical phenomena can be described by a propagating front. A front can for example describe an interface between different objects or fluids (multi-phase flow) [38], a shock wave [32], or the arrival of a wave [44]. Front tracking solvers are used in applications as diverse as mesh generation [43], optimal path planning [31], cardiac activation time estimations [59], and segmentation of images [27]. There are several methods to describe an evolving front. In this thesis the front is assumed monotonically expanding, like a wildfire spreading only to unburned grounds. Monotonically expanding fronts can be entirely described by the time of arrival of the front to all points in the domain. A motivational derivation of a differential equation describing monotonic front propagation is given below.

Let $T(\mathbf{x})$ be the time of arrival of a front to point $\mathbf{x}$, and F the speed of the front in its normal direction. The normal to the front, $\mathbf{n}$, points to regions not yet reached by the front, and is defined as

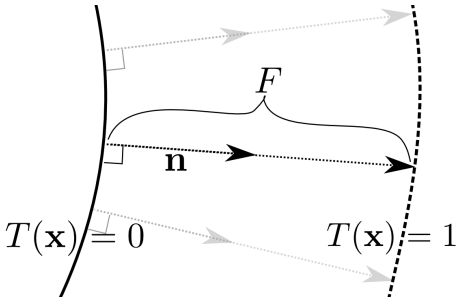


Figure 4: The solid and dashed black lines show the position of the front at time 0 and 1, respectively. The front moves to the right, with a speed of F in the direction orthogonal to the front, $\mathbf{n}$.

$\mathbf{n} = \nabla T / \|\nabla T\|$ where $\|\cdot\|$ is the Euclidean norm. Figure 4 gives a two-dimensional illustration of this principle. We assume that the velocity is well behaved locally, that is, it can be modelled as locally constant in a given direction. If the time of arrival at point $\mathbf{x}$ is known, we seek the time when the front reach point $\mathbf{x} + s\mathbf{n}$ for a small $s > 0$. The time of arrival can be found from the definition of velocity, $\frac{\texttt{time}}{\texttt{distance}} = \frac{1}{|\texttt{velocity}|}$, which translates to

$$\frac{T(\mathbf{x}+s\mathbf{n}) - T(\mathbf{x})}{\|\mathbf{n}s\|} = \frac{1}{F}.$$

Notice that the left hand side of this equation converge to $\mathbf{n} \cdot \nabla T(\mathbf{x})$ as s approaches zero. Using the definition of $\mathbf{n}$, we reach the partial differential equation

$$F\|\nabla T\| = 1. \tag{1}$$

In order to get a solvable system, the solution must be initially known at some points. The set of initially known points is here denoted Γ, and their values are given by the function g. If g is a constant t_0, then Γ is the shape of the front at time t_0.

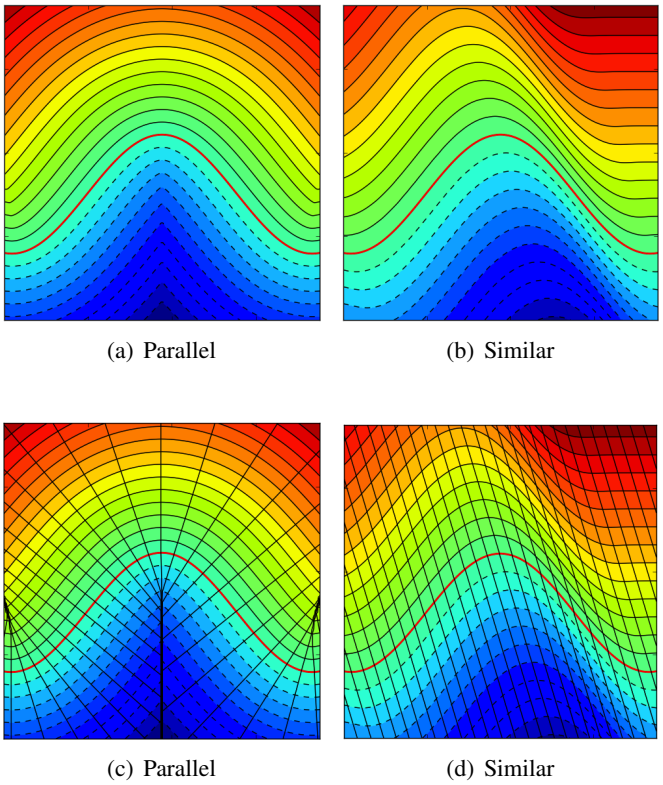
The eikonal equation. In many physical problems the velocity does not change with direction and the problem is referred to as *isotropic*. In the *eikonal* equation, F may change with location but cannot depend on the orientation of ∇T,

$$\begin{aligned} F(\mathbf{x})\|\nabla T(\mathbf{x})\| &= 1, \\ T(\mathbf{x}) &= g(\mathbf{x}) \; \forall \; \mathbf{x} \in \Gamma. \end{aligned} \tag{2}$$

This equation arises in a vast range of applications [52], and can be derived as a high frequency approximation of the wave equation. The eikonal equation is discussed in more depth in the papers to follow in this thesis, and in particular in paper I. A more general framework describing front propagations is that of static Hamilton-Jacobi equations.

Static Hamilton-Jacobi equations. Consider a wildfire spreading through an area with a wind blowing steadily from the south. The wind transports hot air and burning material, causing the fire to spread faster towards north than towards other directions. In this example F changes with the orientation of the front, ∇T, and the problem is said to be *anisotropic*. The formulation of anisotropic problem can vary considerably between applications and a more general framework is therefore considered. All front propagations considered in this work are modelled with static Hamilton-Jacobi equations formulated as

$$\begin{aligned} H(\mathbf{x}, \nabla T(\mathbf{x})) &= 1, \\ T(\mathbf{x}) &= g(\mathbf{x}) \; \forall \; \mathbf{x} \in \Gamma. \end{aligned} \tag{3}$$



(a) Parallel (b) Similar

(c) Parallel (d) Similar

Figure 5: Two different fold types simulated from the same boundary horizon, shown in bright red. Dashed (solid) black lines show a selection of horizons below (above) the reference horizon. (a) Fold of parallel class, with superimposed dip isogons in (c). (b) Fold of similar class, with superimposed dip isogons in (d).

Hamilton-Jacobi equations are known to have multiple valued solutions. For our application, we search for the first time of arrival, which corresponds to the viscosity solution of the static Hamilton-Jacobi equation [52]. Multiple-valued solutions are relevant in some applications, for instance in seismic forward modelling [49]. Still, multiple-valued solutions are beyond the scope of this thesis.

Certain kinds of geological structures cannot be restored using an eikonal formulation. In the following section we present different fold restorations and the static Hamilton-Jacobi equations suitable for dealing with such types of problems.

4 Fold simulations by front propagations

The effect of many geological processes can be reversed through a geometrical transformation of the model. To perform such geometrical mappings, a generalised concept of distance is needed to relate objects in the current and the restored spaces [42]. The boundary condition when simulating folding is given as $T = 0$ on an initially given reference horizon Γ. Other folded horizons are given implicitly as isosurfaces of T (isocurves in two-dimensions). If visualised, the isosurfaces to the distance field appears as horizons in the fold being restored. Layers of sediments are folded differently due to differences in the forces acting on the geological volume. To restore folds correctly, the restoration must be performed on basis of distance fields matching the folding type.

Folds can be classified depending on the changes in curvature between the inner and outer horizons to folded layers of rocks [48]. The *strike* of a folded layer is defined as the intersection of a folded

horizon with a horizontal plane, and the *dip* show the steepest angle of the horizon orthogonal to the strike. *Dip isogons* are lines connecting points of same dip at different layers, that is points on different layers with identical angles to a horizontal plane. Geologists often take measurements of dip and strike when investigating outcrops, and use these measurements to extrapolate the folds beneath the visible ground. Figure 5 shows two different fold simulations from the same reference horizon. The fold in 5(a) is said to be *parallel*, and the fold in 5(b) is *similar* [25, 48]. To further visualise the difference between similar and parallel folds, a selection of dip isogons are shown as black lines in figures 5(c) and (d).

In similar folding, all layers of sediments have identical shape. Since every layer is identical, every horizon in the fold can be modelled by advecting the reference horizon through the domain. The corresponding equation is given by a pure advection formulation,

$$\begin{aligned} \psi(\mathbf{a}\cdot\nabla T) &= 1, \\ T(\mathbf{x}) &= 0 \ \ \forall \mathbf{x}\in\Gamma, \end{aligned} \tag{4}$$

where ψ is a constant advection speed and $\mathbf{a}$ is a unit vector pointing in the same direction as the dip isogons. In folding terminology, $\mathbf{a}$ lies in the axial plane and points along the symmetry line of the fold [25, 26]. The curvature does not change between horizons, as seen from the parallel dip isogons.

Parallel folds are simulated by solving the eikonal equation with a constant velocity. If we choose $F \equiv 1$, a horizon in the fold is at a constant Euclidean distance from the reference horizon. The dip isogons show the minimal Euclidean distance-path to the reference horizon in parallel folding. In parallel folding, every sediment layer has the same thickness unlike the layers of similar folds. Notice that the dip isogons spread out away from the center of the fold, that is, the curvature is decreasing away from the reference horizon.

All other types of folds can be described by combining equations (2) and (4) to the following form

$$F\|\nabla T(\mathbf{x})\| + \psi(\mathbf{a}\cdot\nabla T(\mathbf{x})) = 1, \tag{5}$$

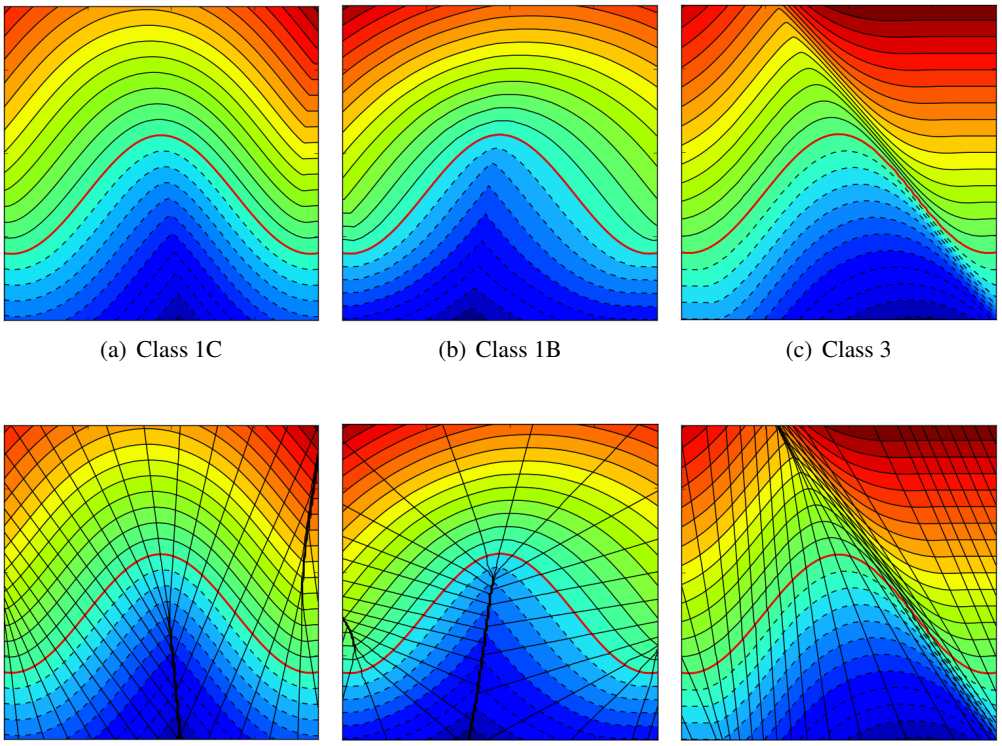
$$T(\mathbf{x}) = 0 \ \ \forall \mathbf{x}\in\Gamma. \tag{6}$$

This equation is of static Hamilton-Jacobi form, as defined in (3). Different folding classes are simulated by varying the size and sign of the speed components F, ψ, and the direction $\mathbf{a}$. The remaining folding classes 1C, 1B, and 3 are shown in figure 6, with and without dip isogons. The signs of the constants ψ and F are mentioned in the caption for the different fold types. Different parameters result in different layer-thickness variations, and the type of folding therefore influence the distribution of rock-properties in a restoration step. Combinations of folding types can be simulated by letting F, ψ and $\mathbf{a}$ change with location. A condition for existence of a solution is that $F\mathbf{n} + \psi\mathbf{a} > 0$. For a through derivation of the mathematical model, and further details, see [25, 26].

5 Solution algorithms

There are several approaches to approximate solutions of equations describing a propagating front. In this thesis we consider methods for a discrete domain of nodes in a rectangular grid, with values given for some nodes. More complete algorithmic overviews are given in the papers to follow in this thesis, in particular in paper IV. Most front propagation equations are nonlinear, and the nodal update procedure is expensive in terms of computations. In paper I, three upwind finite difference stencils are compared with regards to accuracy and efficiency [20].

The obvious, but inefficient, approach is to compute new values for all nodes iteratively [23, 51]. Causality tells us that the current position of a front cannot affect earlier positions of the front. Therefore, it is not necessary to compute new values in areas already passed by the front, or in areas



(a) Class 1C (b) Class 1B (c) Class 3

(d) Class 1C (e) Class 1B (f) Class 3

Figure 6: The remaining fold types not shown in figure 5, and the signs of F and ψ that is needed to simulate them. (a) and (d) class 1C ($\psi > 0$, $F > 0$). (b) and (e) class 1B ($\psi < 0$, $F > 0$). (c) and (f) class 3 ($\psi > 0$, $F < 0$).

far ahead of the front. The amount of computations can thus be reduced if nodes are updated in an order corresponding to this causality observation. There are three main groups of algorithms for fast approximations of monotonically expanding fronts.

Front tracking. Instead of computing values for all nodes in the grid, *tracking methods* simulate an evolving front as it passes one node at a time. The current position of the front is described by a set of nodes called the *narrow band* [47, 52]. In every iteration, the node with the smallest value in the narrow band is considered to be passed by the front, and removed from the narrow band. Nodes close to the passed node are updated and added to the narrow band, but only if they have not been passed earlier. A node can only be passed by the front once, and the number of "iterations" of tracking methods is equal to the number of unknown nodes. The most well known tracking methods is the *fast marching method* (FMM) [52] and its anisotropic extension, the *ordered upwind method* [53]. Since only one node is passed at a time, the methods are sequential by construction. In order to find which node is to be passed, the nodes in the narrow band must be sorted, and the computational cost increases as $N \log(N)$, where N is the total number of nodes.

Sweeping. Instead of considering the solution as the arrival time of a front, it can be viewed as a generalised distance to Γ. The distance is not necessarily measured in an Euclidean sense, but may depend on both location and direction. Distances are easily measured in one direction at a time. *Sweeping methods* make use of this observation by updating nodes in one direction at a time [46], in a set of directions spanning the solution space. The number of sweeps needed for convergence depends

on the geometry of the domain since obstacles can force distances to be measured along curved paths. Sweeping methods can be faster than tracking methods on simple examples, but tracking methods are faster when the domain or velocity formulations are nontrivial [23]. Unlike the ordering of updates in tracking methods, the order of nodal updates is entirely predefined in a sweep. The most well known sweeping method is the *fast sweeping method* (FSM) [61]. With some alterations it is possible to attain parallel implementations of sweeping-like methods [60, 62], such as in paper III [22].

Label correcting methods: Algorithms that classify nodes but allow passed nodes to be revisited, are known as *label correcting* [3, 13]. Paper II presents a label correcting method for anisotropic problems [18]. Similar to the narrow band of tracking methods, a list is often used to keep track of nodes to be updated, but the criteria for a node to enter or exit the list differ. Label correcting methods often try to behave similar to tracking methods, but with a relaxed ordering of updates. This makes label correcting methods less stable in performance than tracking methods, but often faster since the list does not need to be sorted, unlike the narrow band in tracking methods. Parallel implementations of label correcting methods are sometimes possible, for example as in the *fast iterative method* (FIM) [28].

It is also possible possible to combine different types of methods. For instance, the *two-scale method* combines tracking and sweeping methods [8]. First, the domain is divided into smaller subdomains. A sweeping method is used within a subdomain, but subdomains are updated in an order determined by a tracking method. A local sweeping method converges in a few iterations, and the resulting method is very fast compared to both traditional tracking and sweeping methods. Very recently a parallel two-scale method was introduced by the same authors [9]. The new algorithms presented in paper IV are independently developed parallel two-scale algorithms, combining label correcting and sweeping methods on different scales [19].

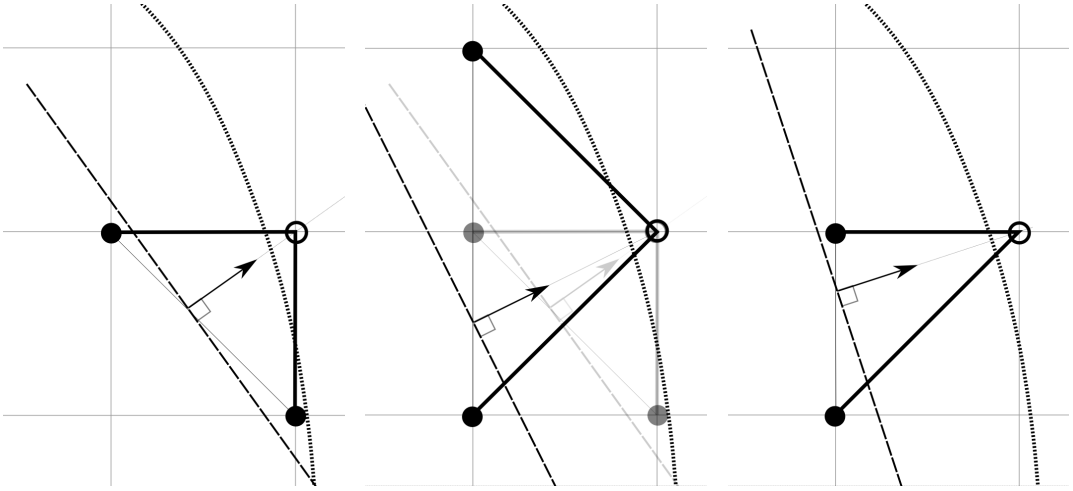
6 Research papers

Restoration in CES is based on simulations of geological folding created with a tracking method [26]. In most two-dimensional restorations, the fold simulations are fast enough to make the software interactive. The simulation time increases rapidly with the grid size in three dimensions, and fold simulations on large grids can stall the CES software for several minutes. In this thesis, different aspects of front propagation relevant for CES are investigated, as outlined below. The research papers I – III are reproductions of published material with minor cosmetic alterations. Paper IV is a technical report from which journal papers will be extracted. Since this thesis is a collection of papers, much of the material from this introduction is repeated in the papers.

6.1 Paper I: Accuracy and efficiency of stencils for the eikonal equation in earth modelling

There are several different approaches to discretise front propagation simulations, including both finite element [4, 33] and finite difference [16, 30, 52] discretizations. The most common approach is upwind finite difference schemes [52], since upwind stencils are accurate, efficient, and does not introduce much numerical diffusion.

Three different upwind stencils for the eikonal equation in two dimensions are investigated in Paper I [20]. All stencils model a segment of the front as planar, but use nodes configured in different shapes. Figure 7 illustrate the three stencils, from here on referred to as the *Godunov*, *multistencil* and *diagonal* shapes, respectively. The stencils are derived by modelling a segment of the front (dashed lines), resulting in an improved condition for accepting a nodal approximation in the multistencil case. In previous works, the upwind conditions for the multistencil could yield imaginary arrival times [24].



(a) Godunov shape (b) Multistencil shapes (c) Diagonal shape

Figure 7: Three stencil shapes. Black circles show the nodes used to update the node marked with an empty circle. The dashed line shows the modelled planar front segment, and the dotted curved line show an isocurve of the front. (a) The Godunov stencil shape that is used in a majority of research papers. (b) The multistencil approach where the Godunov and a rotated stencil are used. (c) The diagonal stencil shape.

Our improved acceptance condition assures that only real solutions are attained. We also introduce a methodology for deriving higher order upwind stencils of any shape, by means of modelling the front. Second order upwind stencils increase the accuracy of the solutions significantly [45, 50]. We present a stable second order diagonal stencil derived using the proposed methodology.

All three stencils were implemented in the tracking solver of CES, in both first and second order versions. The stencils are evaluated in terms of accuracy, efficiency and convergence on several examples with analytical solutions, including a fold simulation example shown in figure 8. Our numerical experiments show that the multistencil is often more accurate than the Godunov stencil. However, the increase in accuracy is expensive in terms of increased computing time. The diagonal shape is the most accurate and fastest, in both first and second order versions.

The idea, derivations, numerical experiments, and stencil implementations were performed by Gillberg. The tracking solver is used with courtesy of Kalkulo, and has been implemented by Hjelle. Most of the article was written by Gillberg, Bruaset wrote the introduction, and the scientific presentation was greatly enhanced by both Bruaset and Hjelle.

6.2 Paper II: A semi-ordered fast iterative method (SOFI) for monotone front propagation in simulations of geological folding

The solution methods described so far all search for values on at a known set of nodes, however, a moving front can also be described by Lagrangian methods. In the Lagrangian approach, the front is described by a set of marker particles that travel from Γ throughout the domain along *rays*, which are also known as ray paths or characteristic curves. When simulating folds, the dip isogons coincides with rays. The solution between rays can be estimated by interpolation [7, 37]. The interpolation quality worsens if the particles become sparse, but new particles can be introduced to fill the gaps [58].

Anisotropic front tracking solvers are difficult to implement since the stencil shapes are changed dynamically in the algorithm, and some algorithms are not applicable on certain problem formulations [6]. On large grids, the sorting needed in tracking methods also becomes a bottleneck [29]. Label correcting methods, on the other hand, require no strict sorting procedure [13], and some label correcting methods

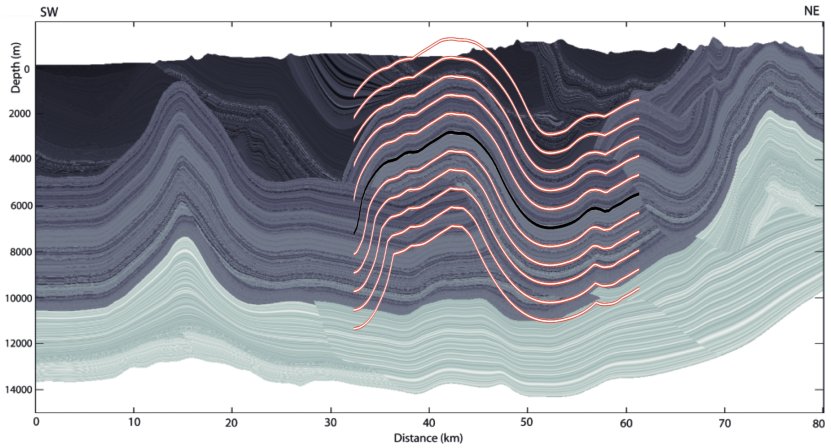


Figure 8: A simulated parallel fold in a region of the Zagros fold belt in Iran [20]. Isocurves of a simulated fold have been superimposed on a visualisation of acoustic impedance data, shown here with courtesy of [2].

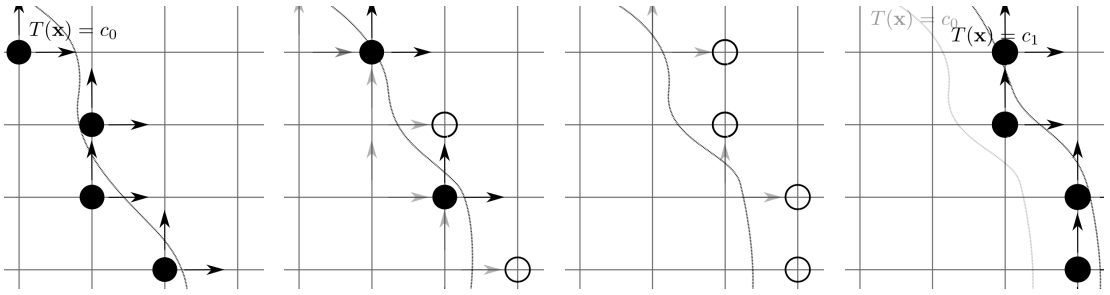
have been shown to be directly applicable to anisotropic problems [27]. A drawback of label correcting methods is that their performance depends on the problem formulation, unlike that of tracking methods. The *two-queue* method enforces a heuristic sorting of updates that reduces the variation in performance, but the method is only applicable to isotropic problems [3].

Paper II introduces a new label correcting algorithm that is applicable to anisotropic problems, the *semi ordered fast iterative method* (SOFI) [18]. SOFI is designed in a way that interpret nodes on the grid as Lagrangian particles. The following algorithmic steps are visualised in figures 9(a) to (d). Nodes labelled *active* are used as marker particles to push the front forward by computing new values for nodes in their vicinity, see figure 9(a). Nodes that lie close to, but ahead of, active nodes are labelled *paused*. SOFI enforces an ordering of updates by using two lists of nodes to describe the front; the *Active* and *Paused* lists, containing active and paused nodes respectively. A node that receives a new value is added to either the Active or Paused list. If the new value is lower than a *cut-off value*, the node is added to the Active list, otherwise to the Paused list, see figure 9(b). When the Active list is empty, as in figure 9(c), the two lists are swapped and the procedure is repeated, see figure (d).

Both the Godunov and the diagonal stencils are used in several numerical experiments. The cut-off value is based on the average value of all active nodes to assure a good distribution of active and paused nodes. The enforced ordering significantly decreases the computing time in most examples, and SOFI is considerably faster when compared to the CES tracking solver on isotropic examples (FMM). During the numerical experiments we noticed that the computing time increases if the ordering is too strict in problems with strong anisotropic influence. Anisotropy can create dependencies of solution estimates from larger valued nodes to smaller valued nodes, and a direct interpretation of the causality principle at the nodal level is then not valid. A too strict ordering therefore increases the number of required iterations.

6.3 Paper III: A new parallel 3D front propagation algorithm for fast simulation of geological folds

Parallel implementations of FSM are possible by sweeping the domain in different directions simultaneously, or sweeping different parts of the domain simultaneously [62]. Implementations of these suggestions can make FSM solvers a couple of times faster [34]. The performance gain is limited since only a few processors can be used in parallel. The *parallel marching method* (PMM) [60] is a sweeping method that sweeps in slightly tilted directions, and use the diagonal stencil to remove the



(a) Only active (b) Active and Paused (c) No active left (d) New iteration

Figure 9: Illustration of the SOFI method on an eikonal case when using Godunov stencils. The dotted line shows the current cut-off c_0-isocurve of the solution. Black circles are active nodes and empty circles are paused nodes. Arrows point from active nodes to nodes being updated. (a) Active nodes are used to update downwind-neighbour nodes. (b) All nodes with new solution value larger than c_0 have been added to the Paused list, and the ones with smaller values to the Active list. (c) After one more iteration of the Active list no more nodes are active. (d) A new cut-off value, c_1, is created and the Active and Paused lists are swapped before the process is restarted.

dependency between two neighbouring nodes. An entire line of nodes in the two dimensional domain can thus be computed simultaneously, and PMM scales well on multicore computers and on GPUs.

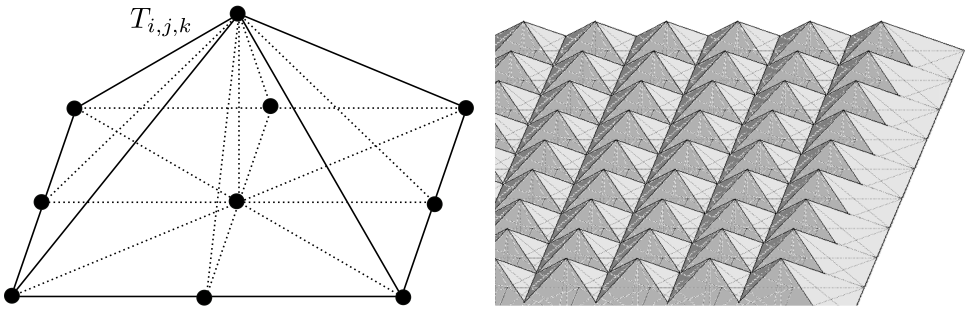
Paper III introduces a three-dimensional version of PMM with the *three-dimensional parallel marching method* (3D PMM) [22]. The stencil must have a shape reminding of a pyramid, as visualised in figure 10(a). The parallel computing possibilities is higher in three dimensions than in two dimensions, since an entire surface of nodes can be computed in parallel, as shown in figure 10(b). Paper III also introduces a new discretization approach for the anisotropic folding equation. Because of the simplicity of 3D PMM, sequential annotated C code can be automatically ported to Nvidia GPUs using the free source code translator Mint [56]. To our knowledge, 3D PMM is only the second front propagation method designed for three-dimensional front propagation problems on GPUs, after the *block-FIM* [28]. Both multicore and GPU implementations were investigated for an example of anisotropic folding. The CPU implementation using OpenMP scales close to linearly with the number of cores on a node with two twelve-core AMD "Magny-Cours" 2.1 GHz processors. The Mint-translated CUDA code on a GeForce GTX 590 GPU is several times faster than all 24 CPU cores working together. In our numerical experiments we noticed that single precision and double precision solutions are practically identical. On an anisotropic folding case with 400^3 nodes, the computing time is 25 seconds on the GPU, and 178 seconds on the CPU, in single precision.

The concepts, the numerical stencil for the anisotropic case, and implementation were developed by Gillberg. Sourouri was at the time a master student of Gillberg, and helped with profiling and numerical experiments. Most of the article was written by Gillberg, with contributions by Cai and Sourouri. Sourouri presented the work at the ICCS 2012 conference. Early results of the work were presented at the EAGE conference in Copenhagen [21] by Gillberg. For a detailed analysis of 3D PMM, see the master's thesis of Sourouri [54], also published at gpuscience.com[1].

6.4 Paper IV: Parallel solvers for static Hamilton-Jacobi equations in three dimensions

A drawback of sweeping-like methods is that the entire domain sometimes must be swept repeatedly until convergence. The 3D PMM method may need several sweeps for full convergence in some

[1] http://gpuscience.com/articles/a-parallel-front-propagation-method-simulating-geological-folds-on-parallel-architectures



(a) Only active (b) Active and Paused

Figure 10: (a) The pyramid stencil used in 3D PMM. Nodal values from the pyramid bottom are used to update the value of node $T_{i,j,k}$. (b) A layer of pyramid stencils, of which all top nodes can be updated in parallel since none of the computed values affect the others.

anisotropic fold simulations. In order to get efficient solvers, the causality principle must be further taken into consideration in the algorithmic design. Paper IV introduces three algorithms designed for parallel implementations that make better use of causality. These algorithms are influenced by 3D PMM, the parallel FSM variants, and the two-scale method. The domain is divided into subdomains that are padded with a layer of ghost nodes. Ghost nodes are copies of nodes located in neighbouring subdomains. By using ghost nodes, any two subdomains can be computed simultaneously with no memory interference. Any algorithm can be used to compute solution values within a subdomain, but in the investigated implementations we used a 3D PMM solver. To prevent unnecessary computations of subdomains, a compute lock is introduced at the subdomain level. All subdomains are locked for computations until they receive a new value on a node. After updating the nodal values in a subdomain, the subdomain's compute lock is again locked for further computations. The algorithms can be described with the task schedule solver shown in algorithm 6.1.

Algorithm 6.1: SOLVER()

comment: Generic driver for the algorithms in paper IV.

```
BUILDSCHEDULE(Schedule)
while Schedule is not empty
do { while Repeat condition not fulfilled
     do { COMPUTESCHEDULE(Schedule)
          SYNCFROMSCHEDULE(Schedule)
   BUILDSCHEDULE(Schedule)
```

A set of subdomains is collected in a schedule, and all scheduled subdomains are processed in parallel. After the scheduled subdomains have been computed, the values of computed nodes that exist in several memory locations must be updated. This synchronization of nodal values is done in a synchronization procedure. The three algorithms create their schedules differently but are otherwise very similar. A schedule is reused to resolve internal dependencies between subdomains in the schedule. The *two scale parallel marching method* (TPM) updates layers of subdomains, similar to the nodes in the 3D PMM method. The *list of active subdomains method* (LAS) adds all subdomains with an open compute lock to a list that acts as the schedule. The *semi ordered list of active subdomains method* (SOLAS) enforces a heuristic ordering of the subdomains on top of LAS.

Since the methods use a local sweeping scheme inside the subdomains, they can be viewed as parallel variations of the two-scale method. TPM can also be connected to the parallel FSM variant where subdomains are swept independently. All methods are of label-correcting type, and LAS have some characteristics similar to the massive marching and FIM methods [13, 28]. The idea of SOLAS is similar to the SOFI methodology in paper II [18]. Since the subdomains are rather large, the extra ordering of SOLAS is not very important, and the difference in performance between LAS and SOLAS is small.

Paper IV investigates the algorithms empirically for several fold simulations and synthetic examples, performed on typical workstations. Two different multicore CPUs are tested, as well as one laptop GPU and one desktop GPU. Paper IV is published as a technical report, and will be reformatted and submitted as a journal publication.

The algorithmic design and implementations were contributed by Gillberg. The majority of the report was written by Gillberg and Bruaset, with some contributions from Sourouri and Hjelle. Sourouri contributed also with profiling, suggestions for code optimisation, and execution of numerical experiments. The tracking solver in CES was developed by Hjelle, and is used here with courtesy of Kalkulo.

For a comparison of performance to 3D PMM we simulated an anisotropic folding example on a grid of 406^3 nodes on a GeForce GTX 590 GPU. The example is similar to the fold simulation in paper III, in which 3D PMM needs 25 seconds using the same GPU. On the similar folding example, TPM needs 15 seconds, and LAS and SOLAS around 4.5 seconds. On the stronger Tesla K20 GPU the simulation times are reduced to 2.5 seconds for LAS and SOLAS, allowing anisotropic folds to be simulated on large grids in a few seconds. These new algorithms makes it possible to use CES interactively even for large, detailed models of complex geology.

Bibliography

1. (2013, July). Top500 supercomputer site. `http://www.top500.org/`.

2. Alaei, B. and S. A. Petersen (2007). Geological modelling and finite difference forward realization of a regional section from the Zagros fold-and-thrust belt. *Petroleum Geoscience 13*(3), 241–251.

3. Bak, S., J. McLaughlin, and D. Renzi (2010, September). Some improvements for the fast sweeping method. *SIAM Journal on Scientific Computing 32*(5), 2853–2874.

4. Bornemann, F. and C. Rasch (2006). Finite-element discretization of static Hamilton-Jacobi equations based on a local variational principle. *Computing and Visualization in Science 9*(2), 57–69.

5. Bruaset, A. M. (2010). *Simula Research Laboratory – by thinking constantly about it*, Chapter Turning Rocks into Knowledge, pp. 553–600. Springer.

6. Cacace, S., E. Cristiani, and M. Falcone (2013). Requiem for local single-pass methods solving stationary Hamilton-Jacobi equations? *arXiv preprint arXiv:1301.6775*, 1–22.

7. Cerveny, V. (2001). *Seismic Ray Theory*. Cambridge University Press.

8. Chacon, A. and A. Vladimirsky (2012). Fast two-scale methods for eikonal equations. *SIAM Journal on Scientific Computing 34*(2), A547–A578.

9. Chacon, A. and A. Vladimirsky (2013). A parallel heap-cell method for eikonal equations. *arXiv preprint arXiv:1306.4743*, 1–31.

10. Chernicoff, S. and D. Whitney (2006). *Geology: An introduction to physical geology* (4 ed.)., Chapter 6, 7. Pearson Prentice Hall.

11. De Santi, M. R., J. L. E. Campos, and L. F. Martha (2002). A finite element approach for geological section reconstruction. In *Proceedings of the 22th Gocad Meeting, Nancy, France*, pp. 1–13.

12. De Santi, M. R., J. L. E. Campos, and L. F. Martha (2003). 3-D geological restoration using a finite element approach. In *Gocad Proceedings: 23th Gocad Meeting, Association Scientifique pour la Geologie et ses Applications*, pp. 1.

13. Dejnozkova, E. and P. Dokladal (2003). A parallel algorithm for solving the eikonal equation. In *Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03). 2003 IEEE International Conference on*, Volume 3, pp. 325 – 328. IEEE.

14. Egan, S., T. Buddin, S. Kane, and G. Williams (1997). Three-dimensional modelling and visualisation in structural geology: New techniques for the restoration and balancing of volumes. In *Proceedings of the 1996 Geoscience Information Group Conference on Geological Visualization, Electronic Geology*, Volume 1, pp. 67–82.

15. Floater, M. S., G. Kós, and M. Reimers (2005, October). Mean value coordinates in 3D. *Computer Aided Geometric Design 22*(7), 623–631.

16. Galbraith, M., R. Kolesar, and Z. Yao (2013). Regular grids travel time calculation - fast marching with an adaptive stencils approach. *geoconvention.com* (1), 1–5.

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

18. Gillberg, T. (2011). A semi-ordered fast iterative method (SOFI) for monotone front propagation in simulations of geological folding. In *MODSIM2011, 19th International Congress on Modelling and Simulation*, pp. 641–647.

19. Gillberg, T., A. M. Bruaset, M. Sourouri, and Ø. Hjelle (2013, July). Parallel solvers for static Hamilton-Jacobi equations in three dimensions. Technical report, Simula School of Research and Education.

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

21. Gillberg, T., Ø. Hjelle, and A. M. Bruaset (2012b). A parallel 3D front propagation algorithm for simulation of geological folding on GPUs. In *EAGE 74th Conference & Exhibition, Extended Abstracts*. EarthDoc.

22. Gillberg, T., M. Sourouri, and X. Cai (2012). A new parallel 3D front propagation algorithm for fast simulation of geological folds. *Procedia Computer Science 9*, 947–955. Proceedings of the International Conference on Computational Science, ICCS 2012.

23. Gremaud, P. A. and C. M. Kuster (2006, December). Computational study of fast methods for the eikonal equation. *SIAM Journal on Scientific Computing 27*(6), 1803–1816.

24. Hassouna, M. S. and A. A. Farag (2007, September). Multistencils fast marching methods: A highly accurate solution to the eikonal equation on cartesian domains. *IEEE Transactions on Pattern Analysis and Machine Intelligence 29*(9), 1563–1574.

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

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

27. Jeong, W.-K., P. T. Fletcher, R. Tao, and R. Whitaker (2007, November – December). 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.

28. Jeong, W.-K. and R. Whitaker (2007). A fast eikonal equation solver for parallel systems. In *SIAM conference on Computational Science and Engineering*.

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

30. Kao, C. Y., S. Osher, and J. Qian (2004, May). Lax-Friedrichs sweeping scheme for static Hamilton-Jacobi equations. *Journal of Computational Physics 196*(1), 367–391.

31. Kimmel, R. and J. A. Sethian (2001). Optimal algorithm for shape from shading and path planning. *Journal of Mathematical Imaging and Vision 14*, 237–244.

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

33. Li, F., C. C.-W. Shu, Y. Y.-T. Zhang, and H. Zhao (2008, September). A second order discontinuous Galerkin fast sweeping method for Eikonal equations. *Journal of Computational Physics 227*(17), 8191–8208.

34. Li, S., K. Mueller, M. Jackowski, D. Dione, and L. Staib (2008). Physical-space refraction-corrected transmission ultrasound computed tomography made computationally practical. In *Medical Image Computing and Computer-Assisted Intervention - MICCAI 2008*, Volume 5242 of *Lecture Notes in Computer Science*, pp. 280–288. Springer Berlin / Heidelberg.

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

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

37. Moser, T. J. (1994, July). Migration using the shortest-path method. *Geophysics 59*(7), 1110–1120.

38. Natvig, J. R. and K.-A. Lie (2008). Fast computation of multiphase flow in porous media by implicit discontinuous galerkin schemes with optimal ordering of elements. *Journal of Computational Physics 227*(24), 10108 – 10124.

39. Petersen, S. A. (1999). Compound modelling, a geological approach to the construction of shared earth models. *EAGE 61th Conference & Exhibition, Extended Abstracts*.

40. Petersen, S. A. and Ø. Hjelle (2008). Earth recursion, an important component in shared earth model builders. *EAGE 70th Conference & Exhibition, Extended Abstracts*.

41. Petersen, S. A., Ø. Hjelle, S. Hustoft, and M. Haubiers (2012). Process based data-restoration and model-reconstruction workflow for seismic interpretation and model building. In *EAGE 74th Conference & Exhibition, Extended Abstracts*.

42. Petersen, S. A., Ø. Hjelle, and S. L. Jensen (2007). Earth modelling using distance fields derived by fast marching. *EAGE 69th Conference & Exhibition, Extended Abstracts*.

43. Peyré, G. and L. D. Cohen (2006). Geodesic remeshing using front propagation. *International Journal of Computer Vision 69*(1), 145–156.

44. Podvin, P. and I. Lecomte (1991). Finite difference computation of traveltimes in very contrasted velocity models: a massively parallel approach and its associated tools. *Geophysical Journal International 105*, 271–284.

45. Popovici, A. M. and J. A. Sethian (2002). 3-D imaging using higher order fast marching traveltimes. *Geophysics 67*(604, Issue 2), 604 – 609.

46. Qian, J., Y.-T. Zhang, and H.-K. Zhao (2007). A fast sweeping method for static convex Hamilton-Jacobi equations. *Journal of Scientific Computing 31*(1-2), 237–271.

47. Qin, F., Y. Luo, K. B. Olsen, W. Cai, and G. T. Schuster (1992, March). Finite-difference solution of the eikonal equation along expanding wavefronts. *Geophysics 57*(3), 478–487.

48. Ramsay, J. G. (1967). *Folding and fracturing of rocks*. McGraw-Hill, New York and London.

49. Rawlinson, N., J. Hauser, and M. Sambridge (2009). Seismic ray tracing and wavefront tracking in laterally heterogeneous media. *Advances in geophysics 49*, 203–267.

50. Rickett, J. and S. Fomel (1999, April). A second-order fast marching eikonal solver. In *Stanford Exploration Project Report*, Volume 100, pp. 287–292. SEP.

51. Rouy, E. and A. Tourin (1992). A viscosity solutions approach to shape-from-shading. *SIAM J. Numer. Anal. 29*(3), 867–884.

52. Sethian, J. A. (1999). *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science*. Cambridge University Press.

53. Sethian, J. A. and A. Vladimirsky (2003). Ordered upwind methods for static Hamilton-Jacobi equation: Theory and algorithms. *SIAM Journal of Numerical Analysis 41*(1), 325 – 363.

54. Sourouri, M. (2012). A parallel front propagation method: Simulating geological folds on parallel architectures. Master's thesis, University of Oslo.

55. Suter, E., E. Cayeux, A. Escalona, T. Kårstad, and E. H. Vefring (2012). A strategy for effective local updates of the geological structure in an earth model during drilling. In *74th EAGE Conference & Exhibition*.

56. Unat, D., X. Cai, and S. Baden (2011). Mint: Realizing CUDA performance in 3D stencil methods with annotated C. In *Proceedings of the 25th International Conference on Supercomputing (ICS'11)*, ICS '11, New York, NY, USA, pp. 214–224. ACM.

57. Vidal-Royo, O., N. Cardozo, J. A. Muñoz, S. Hardy, and L. Maerten (2012). Multiple mechanisms driving detachment folding as deduced from 3D reconstruction and geomechanical restoration: the Pico del Águila anticline (External Sierras, Southern Pyrenees). *Basin Research 24*(3), 295–313.

58. Vinje, V., E. Iversen, and H. Gjøystdal (1993). Traveltime and amplitude estimation using wavefront construction. *Geophysics 58*(8), 1157–1166.

59. Wallman, M., N. P. Smith, and B. Rodriguez (2012, June). A comparative study of graph-based, eikonal, and monodomain simulations for the estimation of cardiac activation times. *IEEE transactions on biomedical engineering 59*(6), 1739–1748.

60. Weber, O., Y. S. Devir, A. M. Bronstein, M. M. Bronstein, and R. Kimmel (2008). Parallel algorithms for approximation of distance maps on parametric surfaces. *ACM Transactions on Graphics (TOG) 27*(4), 104:1–104:16.

61. Zhao, H.-K. (2004, May). A fast sweeping method for eikonal equations. *Mathematics of Computation 74*(250), 603–627.

62. Zhao, H.-K. (2007). Parallel implementations of the fast sweeping method. *Journal of Computational Mathematics 25*(4), 421 – 429.

Paper I:
Accuracy and efficiency of stencils for the eikonal equation in earth modelling

Accuracy and efficiency of stencils for the eikonal equation in earth modelling

Tor Gillberg [1,2], **Øyvind Hjelle** [2], **Are Magnus Bruaset** [3,4]

[1] Computational Geosciences, Simula Research Laboratory,
P. O. Box 134, N-1325 Lysaker, Norway

[2] Kalkulo AS,
P. O. Box 134, N-1325 Lysaker, Norway

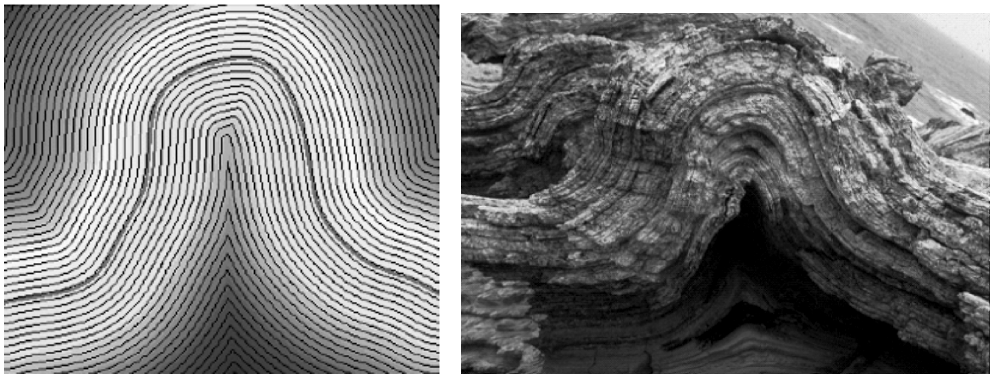
[3] Center for Biomedical Computing, Simula Research Laboratory,
P. O. Box 134, N-1325 Lysaker, Norway

[4] Department of Informatics, University of Oslo,
P. O. Box 1080 Blindern, N-0316 Oslo, Norway

Computational Geosciences 16, number 4 (2012), pages 933-952.

Abstract:

Motivated by the needs for creating fast and accurate models of complex geological scenarios, accuracy and efficiency of three stencils for the isotropic eikonal equation on rectangular grids are evaluated using a Fast Marching implementation. The stencils are derived by direct modelling of the wave front, resulting in new and valuable insight in terms of improved upwind and causality conditions. After introducing a method for generalising first order upwind stencils to higher order, a new second order *Diagonal* stencil is presented. Similarly to the Multistencils Fast Marching approach, the Diagonal stencil makes use of nodes in the diagonal directions, whereas the traditional Godunov stencil uses solely edge-connected neighbours. The Diagonal stencil uses nodes close to each other, reaching upwind, to get a more accurate estimate of the angle of incidence of the arriving wave front. Although the stencils are evaluated in a Fast Marching setting, they can be adapted to other efficient eikonal solvers. All first and second order stencils are evaluated in a range of tests. The first test case models a folded structure from the Zagros fold belt in Iran. The other test cases are constructed to investigate specific properties of the examined stencils. The numerical investigation considers convergence rates and CPU times for non-constant and constant speed first-arrival computations. In conclusion, the Diagonal stencil is the most efficient and accurate of the three alternatives.



(a) Compound model fold (b) Observation from an outcrop

Figure 1: A folded structure computed by the algorithms used in the compound model builder, and an observed folded structure in an outcrop

1 Introduction

Traditional construction of computer-based geological models is a time-consuming process, in which the geoscientists must make a series of assumptions and data interpretations that can dramatically impact the validity of the resulting model. Naturally, the inherent difficulty of such modelling increases the more complex geological scenarios one encounters. In the oil and gas industry, geological modelling has grown in complexity as a result of the global competition in hydrocarbon exploration, which continuously forces the modellers out of their scientific comfort zone as they survey prospective fields world-wide.

To deliver better models faster, considerable research and development efforts are spent on new paradigms that allow models to be more easily extended and edited [25, 26] than those created by traditional workflows. Compound Modelling (CM) [27–30] is one particular technology of this kind, applicable in two and three spatial dimensions. The basic concept of the CM technology is to describe geology as the realization of a series of geological events and processes along a geological timeline. The resulting model can represent geological structures of almost arbitrary complexity, including intricate relationships between horizons, faults, and physical installations such as wells.

The compound model builder creates subsurface geometries by combining seismic interpretations, well observations and other types of hard data with geological experience and intuition represented by a transient process description. In addition, the builder populates this geometry with physical properties based on computed distances between each grid cell and the physical objects present in the model. As explained in [8], the CM approach relies heavily on efficient and accurate calculation of distance fields. In general, this calculation consists of solving the static Hamilton-Jacobi systems derived and presented in [13]. However, in this paper we restrict our scope to methods for simulating parallel folds through solution of the boundary value problem

$$F(x)\|\nabla T(x)\| = 1, \qquad T(x) = 0 \;\forall\, x \in \Gamma, \tag{1}$$

known as the eikonal equation. Here Γ denotes the initial position of a wave travelling with a speed F in the normal direction, and $\|\cdot\|$ the Euclidian distance. The solution $T(x)$ is the wave's time of arrival to position x. Figure 1 shows a folded structure as modelled in the CM builder, and an observation from an outcrop. In the CM context, the Fast Marching method [38] is the preferred strategy for solving (1), although there are alternative solution algorithms [10]. Both the Fast Marching method and its competitors are in need of an underlying discretization, typically based on upwind finite differences.

In order for the CM software to function interactively, the folded structures must be modelled by fast and accurate computations. Several discretizations are suggested in the literature, but there seems to be no available comparison of the resulting stencils with respect to accuracy and efficiency. In this paper we report on such comparison of three upwind finite difference stencils with different shapes. The comparison focuses on a geological scenario previously modelled by the CM builder [1]. This scenario is a section from the Zagros fold belt in Iran, which stems from the collision between the Eurasian and Arabian tectonic plates. In addition to the Zagros field test, we report on several other numerical experiments providing information about properties of the examined stencils. According to the numerical experiments reported in Section 4, the stencils show consistent behaviour across geological and artificial test cases.

In addition to the numerical comparisons, we present a method for extending stencils of different shapes to higher order by direct modelling of the wavefront. Using this method, we present a new second order stencil that turns out to be both faster and more accurate than the other examined stencils.

1.1 Applications of the eikonal equation

The eikonal equation appears in a wide variety of research fields and applications. In earth science fast solutions are needed in forward modelling of seismic data [18, 36], in descriptions of complex folding regimes [8, 13], and in reservoir simulations [7]. Moreover, the eikonal equation is often solved in computer visualisation applications, such as segmentation of images [3, 23], optimal path planning and computation of geodesic distances [4, 21, 43].

The eikonal equation has many applications in seismic processing [37]. Instead of computing traveltimes along rays, the entire first arrival traveltime field can be simulated by solving the eikonal equation [35, 41]. Eikonal solvers automatically extrapolate the wavefront into shadow zones, which are areas that ray methods find difficult to image. The fact that only first arrival traveltimes are computed can be seen as a weakness of eikonal solvers, because later arrivals are often important for high-quality imaging [22]. However, by repeating the simulation from reflecting surfaces, multiple reflected traveltimes can be found using eikonal solvers [14, 36]. In order to compute amplitudes of the seismic waves, the solution must be differentiated in postprocessing of the computed traveltime field. This poses high requirements of the accuracy of the computed traveltimes. Traditional first order stencils are not accurate enough to provide a reliable amplitude calculation [37]. Similarly, as for seismic amplitude computations, the solution gradient is used in postprocesses of the simulated folds in the CM software [8]. In order to achieve a reliable gradient of the solution, high accuracy of the solution method is crucial and several improvements have been suggested for the discretization of the eikonal equation. These include use of spherical coordinates [2], general triangulations [43], local mesh refinements [36], assumption of the curvature of the arriving wavefront [6, 41], and discontinuous Galerkin discretizations [44]. All mentioned improvements have been shown to decrease errors at the cost of algorithmic simplicity and increase in computational time. However, within the family of fast upwind finite difference stencils there are alternative stencil shapes that also affect the accuracy of the solution.

The traditional stencil on quadrilateral grids, hereafter called the *Standard* stencil, use only nodes connected through edges [38]. Therefore, the errors created from this stencil are especially large in diagonal directions. A simple method to increase accuracy while maintaining the algorithmic simplicity is to extend the upwind finite difference based stencils to higher order. If second order schemes are used, the diagonal errors are significantly lowered [33, 37]. Another approach to remedy these large errors is to include also diagonal nodes in the stencil update. This can be done by applying an additional stencil that uses only edge connected nodes. This is the *Multistencils* approach [12] (a similar multiple stencil approach was taken earlier in [31]). Another option is to use both edge connected and diagonally

connected nodes in one stencil, forming a stencil we refer to as *Diagonal*. Such a stencil has been suggested in [9, 19, 42], but has been limited to first order only. To our knowledge, no comparison of the accuracy and computational cost of these stencils have been documented. The numerical experiements in Section 4 indicate that the Diagonal shape is the most accurate and fastest of the three choices.

1.2 Efficient numerical approximations

Considering the eikonal equation (1), an alternative interpretation of $T(x)$ is as the minimal time of travel from x to Γ with speed F. The eikonal equation is isotropic in the sense that the speed is independent of direction. One possible interpretation of the equation is as the statement $speed \cdot \frac{time}{distance} = 1$.

Generally, the approximation methods are classified in two groups: front tracking (one-pass) and iterative methods. Front tracking methods follow the physical wave as it expands in the discrete domain. Since the wave only passes each node once, only a few operations per node is necessary. However, to know which node is next to be passed, an ordered data structure is needed. This requirement makes parallel implementations difficult. Algorithms of this group includes foremost Fast Marching and Expanding Wavefront methods [34, 38, 40]. The method can be made faster by passing a set of nodes close to the wave front simultaneously [19].

Iterative methods include Fast Sweeping, Fast Iterative, and Expanding-box methods [17, 20, 42, 46]. The efficiency of these methods depends on the geometry of the domain and variations of the velocity. For simple problems, iterative methods are faster than front tracking methods. For complex problems, the preference is reversed [11, 15, 17, 43]. The Fast Iterative method tries to expand the wave simultaneously throughout the domain. As a result, nodes may be passed several times and the method can be implemented on parallel computers and GPUs [16]. By further enforcing a partial ordering of the updates, the methods can be made more stable with respect to geometry and velocity variations [5, 10].

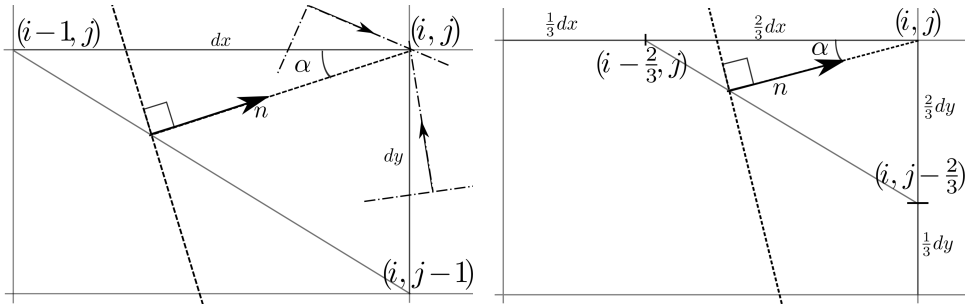
1.3 Outline

The most commonly used first order *Standard* stencil is derived in Section 2.1. In Section 2.2 we present a new method for generalisation of first order upwind stencils to second order by direct modelling of the wave front. After a short discussion on the accuracy of the Standard stencil, the 2D *Multistencils* Fast Marching stencil is derived. The derivation of this stencil by direct modelling of the wave front provides valuable insight, such as the new improved upwind condition for the Multistencils approach. In Section 3 we present the Diagonal stencil, a highly accurate approximation making use of diagonal nodes.

The accuracy and efficiency of all stencils for modelling folds are examined In Section 4.1, where a section of Zagros fold belt in Iran is simulated. To further verify our findings, all numerical stencils are thoroughly tested on six synthetic examples in Section 4.1. The reported CPU times for all stencils are measured when different stencils are applied in the same Fast Marching implementation. The results of these tests, and assessments of the computational complexity of the stencils, are discussed in Sections 5 and 6.

2 Finite difference stencils

In this section, we derive first order stencils by means of direct modelling of the wave front. In Section 2.2, we present a method to generalise these stencils to second order. Every stencil derivation is



(a) Modelled wave moving with constant speed $F_{i,j}$ from node $(i-1,j)$ via $(i,j-1)$ to node (i,j)

(b) Second order derivation illustration

Figure 2: The modelled wave front and local domain used to derive the standard first and second order Standard Godunov schemes

accompanied by a figure that illustrates the setting. A few assumptions are taken in line with the general consensus:

- The grid is quadrilateral and the distances between axis aligned nodes are denoted dx and dy. We use the notation $T_{i,j}$ for the nodal approximation of the solution T in $(x,y) = (i \cdot dx, j \cdot dy)$.
- When solving for an approximation of $T_{i,j}$, the local speed is assumed to be $F_{i,j}$, and hence independent of the direction of the arriving wave front.
- The shape of the arriving wave front is approximated as locally planar. We will refer to such a planar part as a *wave segment*. Since we consider the isotropic eikonal equation (1), the modelled wave segment expands in its normal direction. Consequently, the angle of incidence, that is the angle between the normal and the x- or y- axis, will remain unchanged in the local area of constant speed.

A key observation of these assumptions is that the time that a vertical wave segment needs to travel the distance dx is dx/F. If the wave has an angle of incidence α relative to the x-axis, the time to travel the same axis-aligned distance is $dx\cos\alpha/F$. Given the minimal distance between a modelled wave segment and its arrival point, and the constant speed, we can estimate the first time of arrival of the wave. The minimal distance is the length of the shortest path, which is also known as the ray path.

2.1 The Standard Godunov stencil

Consider the setting given in Figure 2.2(a), where the time of arrival $T_{i,j}$ is to be estimated given arrival times at nodes $(i-1,j)$ and $(i,j-1)$. A front moving horizontally from $(i-1,j)$ to (i,j) needs the time $dx/F_{i,j}$ to travel, but with the current direction of the wave, the time taken is $T_{i,j} - T_{i-1,j}$. Similarly, for a wave moving vertically from $(i,j-1)$ to (i,j) the time of travel is $dy/F_{i,j}$, and the actual time taken is $T_{i,j} - T_{i,j-1}$. These relations give us the following expressions for the angle of incidence α,

$$\frac{T_{i,j} - T_{i-1,j}}{dx/F_{i,j}} = \cos\alpha, \quad \frac{T_{i,j} - T_{i,j-1}}{dy/F_{i,j}} = \sin\alpha. \tag{2}$$

If the angle of incidence is too large, $\alpha > \frac{\pi}{2}$, or too small, $\alpha < 0$, the shortest path lies outside the quadrant (dash-dotted lines in Figure 2.2(a)). The modelled wave is valid only within the element spanned by the included nodes, which explains why only shortest paths within the element should be considered trustworthy. Paths outside the element will be considered when nodes in neighbouring

quadrants are updated. A too large angle indicates that the wave visits node $(i, j-1)$ first, then node (i, j), and finally node $(i-1, j)$. A too small angle similarly indicates that node (i, j) is visited before node $(i, j-1)$. In these cases only one of the neighbours lies upwind of node (i, j), and we have the shortest path along an edge. We formulate these special cases in the following upwind conditions;

$$\begin{aligned} \alpha &\leq 0 \iff T_{i,j-1} > T_{i-1,j} + dx/F_{i,j} \\ &\implies T_{i,j} = T_{i-1,j} + dx/F_{i,j}, \\ \alpha &\geq \frac{\pi}{2} \iff T_{i-1,j} > T_{i,j-1} + dy/F_{i,j} \\ &\implies T_{i,j} = T_{i,j-1} + dy/F_{i,j}. \end{aligned} \tag{3}$$

Adding the squared expressions (2), and combining the restrictions in (3), we end up with the Standard Godunov scheme;

$$\left(\frac{(T_{i,j} - T_{i-1,j})^+}{dx}\right)^2 + \left(\frac{(T_{i,j} - T_{i,j-1})^+}{dy}\right)^2 = \frac{1}{F_{i,j}^2} \tag{4}$$

where $(\cdot)^+$ denotes the positive part of $(\cdot)$. The largest solution gives the arrival time in front of the wave and is thus the one sought for. By directly solving this expression, the largest solution is

$$\begin{aligned} T_{i,j} &= \frac{T_{i-1,j}dy^2 + T_{i,j-1}dx^2}{dx^2 + dy^2} + \\ &\quad \frac{dx\,dy\sqrt{(dx^2+dy^2)/F_{i,j}^2 - (T_{i-1,j} - T_{i,j-1})^2}}{dx^2 + dy^2}. \end{aligned} \tag{5}$$

Node (i, j) belongs to four grid cells that can be used to solve for an arrival time. Since the searched solution is the first arrival time, the smallest of these approximations is the correct one. From the equation above we see that $T_{i,j}$ is decreasing in $T_{i-1,j}$ and $T_{i,j-1}$, and thus the smallest arrival times along the x- and y- axes generate the first time of arrival.

2.2 Second order schemes by direct wave front modelling

Higher order upwind schemes are essentially a forward interpolation of the solution, based on solution values upwind of the point. The second order upwind discretization of a one-dimensional function g is given by

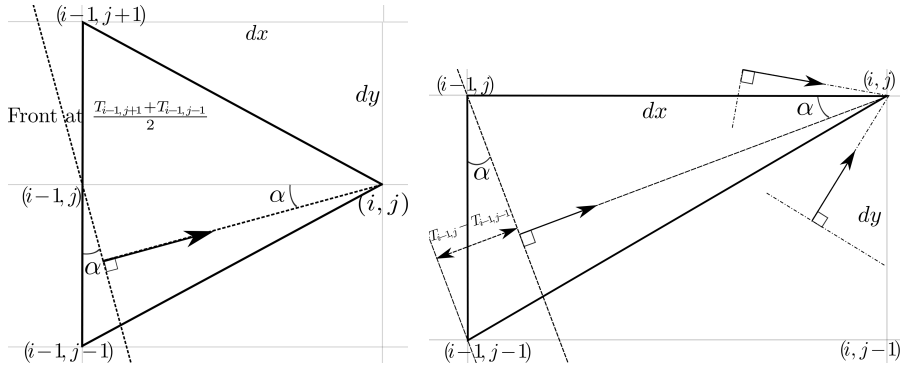
$$\frac{\partial}{\partial x} g_i \approx \frac{3g_i - 4g_{i-1} + g_{i-2}}{2dx} = \frac{g_i - \tilde{g}_{i-2/3}}{(2/3)dx}$$

where $\tilde{g}_{i-2/3} = g_{i-1} + \frac{g_{i-1} - g_{i-2}}{3}$ can be seen as an interpolated value of g at the position $(i-2/3)dx$. Using a second order scheme we get an interpolated value closer to the downwind node i using grid point values upwind of i. This intuitive approach can be used directly on our wave front modelling, as is illustrated in Figure 2(b) for the following second order extension of the standard stencil.

From the above formulation, the wave front time of arrival to $(i-2/3, j)$ is estimated to be $\widetilde{T}_{i-2/3,j} = T_{i-1,j} + \frac{1}{3}(T_{i-1,j} - T_{i-2,j})$, and the time of arrival to node $(i, j-2/3)$ is interpolated as $\widetilde{T}_{i,j-2/3} = T_{i,j-1} + \frac{1}{3}(T_{i,j-1} - T_{i,j-2})$. Since both axes have been scaled equally we end up in the same situation as for the first order scheme. Combining the angle of incidence estimates and bounds we get the following scheme

$$\left(\frac{(T_{i,j} - \widetilde{T}_{i-2/3,j})^+}{(2/3)dx}\right)^2 + \left(\frac{(T_{i,j} - \widetilde{T}_{i,j-2/3})^+}{(2/3)dy}\right)^2 = \frac{1}{F_{i,j}^2}, \tag{6}$$

which is the standard second order upwind Godunov scheme. Notice that this system is very similar to (4) with an explicit solution given by (5). It is important to note that this forward interpolation of the



(a) Derivation of the rotated Multistencils stencil

(b) Derivation of the Diagonal stencil

Figure 3: Derivation of rotated Multistencils and Diagonal stencils

solution is valid only if the more distant node is upwind. A necessary, but not sufficient, condition for $(i-2, j)$ to be upwind of $(i-1, j)$, is that $T_{i-2,j} < T_{i-1,j}$. If this does not hold, the forward interpolation $\widetilde{T}_{i-2/3,j}$ is not valid.

Higher order schemes are reported well worth the increase in computational cost because of the gain in accuracy [33, 45]. An underlying assumption is that the speed is constant in a larger neighbourhood. This assumption explains why second order schemes are not appropriate for applications with non-smooth speed. Similar observations are also seen in the numerical experiments reported in Section 4.1.

2.3 Errors when using the Standard stencil

Consider the problem of computing the distance to a node on a uniform grid, that is $h = dx = dy$ and $F \equiv 1$. First the edge-connected nodes of the starting point will be given the correct solution value, h, from the upwind condition of (3). However, when (5) is solved for the node diagonal to the initial point, it results in the estimated distance $\frac{2+\sqrt{2}}{2}h \approx 1.707h$, which is an overestimation of more than 20%. Only nodes connected via an edge are used to estimate the shape of the wave. Therefore, the angle estimates are especially bad in the diagonal directions [12, 32, 37]. Errors are large for wave fronts with high curvature, since then the planar wave front approximation gives a bad resolution, especially so when only edge connected nodes are used. Second order stencils estimate the angle of incidence closer to the node and are therefore more accurate.

Another way to decrease the errors is to include diagonal points in the stencil. The Multistencils Fast Marching method [12] is one way to do so, and is described in the following section.

2.4 Multistencils Fast Marching

The Multistencils Fast Marching method [12] extends the standard Fast Marching method by using an additional stencil. Both stencils provide approximations, and the smaller one is considered to be the best solution. The additional stencil is created by a rotation of the coordinate axes so that only diagonal nodes are used, as indicated in Figure 3(a). To estimate the angle of incidence in the rotated system, nodes even further apart are used, implying that high curvature still will be modelled inaccurately. Nevertheless, the accuracy is expected to increase compared with the standard stencil since diagonal nodes are incorporated.

The numerical stencils in [12] are derived using directional derivatives, which make them easily extensible to second order. The explicit formulas and numerical examples therein are presented only

on a uniform grid, and the upwind condition may return imaginary arrival times if used, as it allows shortest paths to lie outside the considered element.

The derivation and the numerical experiments in this paper treat also non-uniform grids where $dx \neq dy$. Below we derive the rotated stencil using notations from Figure 3(a), and present a new upwind condition that assures that the ray path is inside the element and that the approximations are real. The underlying assumptions are the same as in the previous section. The angle of incidence is first estimated similarly to the standard case (2),

$$\sin\alpha = F_{i,j}\frac{T_{i-1,j+1} - T_{i-1,j-1}}{2dy}. \tag{7}$$

For the shortest path to lie inside the element spanned by the included nodes we must have $|\sin\alpha| \leq \frac{dy}{\sqrt{dx^2+dy^2}}$. Otherwise, the shortest path is along a side of the element. We present a new upwind condition for the minimal time of arrival by

$$\begin{aligned} F_{i,j}\frac{|T_{i-1,j+1} - T_{i-1,j-1}|}{2dy} \geq \frac{dy}{\sqrt{dx^2+dy^2}} \Longrightarrow \\ T_{i,j} = \min\left(T_{i-1,j+1}, T_{i-1,j-1}\right) + \frac{\sqrt{dx^2+dy^2}}{F_{i,j}}. \end{aligned} \tag{8}$$

Halfway between the $T_{i-1,j+1}$ and $T_{i-1,j-1}$ wave fronts, we are at the solution time $\frac{T_{i-1,j+1}+T_{i-1,j-1}}{2}$, which is our guess of the time of arrival to node $(i-1,j)$. The shortest distance from the $\frac{T_{i-1,j+1}+T_{i-1,j-1}}{2}$ front to (i,j) is $dx\cos\alpha$. Using this relation and applying the trigonometric identity to (7), we get

$$\begin{aligned} T_{i,j} \quad = \quad & \frac{T_{i-1,j+1} + T_{i-1,j-1}}{2} + \\ & dx\sqrt{\frac{1}{F_{i,j}^2} - \frac{(T_{i-1,j+1} - T_{i-1,j-1})^2}{4dy^2}}. \end{aligned} \tag{9}$$

Observe that the solution at node $(i-1,j)$ is not used, but instead interpolated as the average of the solutions above and below. As a consequence, we can expect odd behaviour in the case of high curvature or colliding fronts from above and below. In such cases the estimated traveltime will be to small, leading to errors that propagates with the expanding wavefront (this effect is visible in the shock region of Figures 6(a) and (b)). For a non-uniform grid we do not know which of the rotated stencils gives the smallest time of arrival, and therefore all four new stencils must be solved when the point is updated.

For the second order scheme we use the same technique as presented in Section 2.2. The second order stencil solution is found by replacing the values $T_{i-1,j-1}, T_{i-1,j+1}, dx$ and dy in (9) and (8) with $\widetilde{T}_{i-1,j-1} = \frac{4T_{i-1,j-1}-T_{i-2,j-2}}{3}$, $\widetilde{T}_{i-1,j+1} = \frac{4T_{i-1,j+1}-T_{i-2,j+2}}{3}$, $\widetilde{dx} = (2/3)dx$, and $\widetilde{dy} = (2/3)dy$.

3 Diagonal stencil for improved accuracy

Another approach to include diagonal nodes in the stencils, is to use both diagonal and edge connected nodes. This idea has been discussed in several papers [4, 9, 19, 42]. In [42] a first order scheme using an edge-connected and a diagonal node to increase the angular resolution is used. Similar stencils, but with the restriction $dx = dy$, are given in [4, 24]. With the Diagonal shape, the wave segment is modelled as planar in a smaller neighbourhood compared to the other stencils. By doing so, one gets a

narrower restriction on the angle of incidence, $0 \leq \alpha \leq \arctan \frac{dy}{dx}$, and a more accurate approximation. All the referred methods are of first order accuracy, but in Section 3 a second order stencil using the diagonal nodes is presented. First, we derive the first order stencil and formulate efficient upwind conditions.

The setting is illustrated in Figure 2.3(b), where the arrival time to node (i, j) is to be estimated using given solution values at $(i-1, j)$ and $(i-1, j-1)$. Since the speed is the same in all directions, a wave takes the time $dy/F_{i,j}$ to travel vertically from $(i-1, j-1)$ to $(i-1, j)$. With the modelled angle the actual time needed is $T_{i-1,j} - T_{i-1,j-1}$, which gives the following relation with the angle of incidence

$$\sin\alpha = F_{i,j}\frac{T_{i-1,j} - T_{i-1,j-1}}{dy}. \tag{10}$$

To ensure that the shortest path is inside the modelled area, we bound the angle of incidence. The angle must be positive, and the largest allowed angle must model the ray path along the diagonal edge. Otherwise, the closest path within the element is our best estimate of the arrival time. These restrictions are formulated in the following upwind conditions

$$T_{i-1,j} < T_{i-1,j-1} \implies T_{i,j} = T_{i-1,j} + \frac{dx}{F_{i,j}} \tag{11}$$

$$\frac{T_{i-1,j} - T_{i-1,j-1}}{dy} F_{i,j} > \frac{dy}{\sqrt{dx^2 + dy^2}} \implies$$

$$T_{i,j} = T_{i-1,j-1} + \frac{\sqrt{dx^2 + dy^2}}{F_{i,j}}. \tag{12}$$

For wave fronts upwind of (i, j), the distance between (i, j) and the $T_{i-1,j}$ wave front is $dx\cos\alpha$. Using this relation in combination with (10) and the trigonometric identity gives

$$T_{i,j} = T_{i-1,j} + dx\sqrt{\frac{1}{F_{i,j}^2} - \frac{(T_{i-1,j} - T_{i-1,j-1})^2}{dy^2}}. \tag{13}$$

If we are to update node (i, j) using node $(i-1, j)$, there are two diagonal nodes that can be used. In the expression above, notice that $T_{i,j}$ is decreasing with the value of the diagonal node, that is, a smaller valued diagonal node model the $T_{i-1,j}$ wave front closer to node (i, j). Therefore the diagonal node with smaller time of arrival should be used when choosing stencil to solve for the first time of arrival.

As before, the second order stencil is given by (13) with a change of constants to $\widetilde{T}_{i-1,j} = T_{i-1,j} + \frac{T_{i-1,j} - T_{i-2,j}}{3}$, $\widetilde{T}_{i-1,j-1} = T_{i-1,j-1} + \frac{T_{i-1,j-1} - T_{i-2,j-2}}{3}$, $\widetilde{dx} = \frac{2}{3}dx$, and $\widetilde{dy} = \frac{2}{3}dy$.

We search for upwind stencils in both diagonal directions, and use the smallest interpolated value since it models the wave closest. In the case of upwind stencils in both the diagonal and axis directions, the interpolated constants are used to solve for an arrival time. If we only have a second order stencil in one direction, the first order scheme is used, but the upwind conditions are modified. With a second order stencil only in the x direction, the upwind condition (11) is changed to;

$$T_{i-1,j} < T_{i-1,j-1} \implies T_{i,j} = \widetilde{T}_{i-1,j} + \frac{(2/3)dx}{F_{i,j}}.$$

Similarly, if we only have a second order stencil in the axis-aligned direction, the upwind condition (12)

is changed to

$$\frac{T_{i-1,j}-T_{i-1,j-1}}{dy}F_{i,j} > \frac{dy}{\sqrt{dx^2+dy^2}} \Longrightarrow$$
$$T_{i,j} = \widetilde{T}_{i-1,j-1} + \frac{2/3\sqrt{dx^2+dy^2}}{F_{i,j}}.$$

3.1 A strict causality condition

The causality principle tells us that only nodes with a smaller value can affect the solution at any given node [39]. Following notations in Figure 2.3(b), we derive the following stricter causality principle for the first order Diagonal stencil:

Lemma Diagonal Stencil Causality Conditions

- Edge dependency: $T_{i,j}$ can only depend on $T_{i-1,j}$ if $T_{i,j} > T_{i-1,j} + \frac{dx^2}{F_{i,j}\sqrt{dx^2+dy^2}}$.
- Diagonal dependency: $T_{i,j}$ can only depend on $T_{i-1,j-1}$ if $T_{i,j} \geq T_{i-1,j-1} + \frac{\min(dx,dy)}{F_{i,j}}$.

Proof. The solution at (i,j) is dependent on the neighbouring nodes that model the wave closest to (i,j). From any diagonally connected node we have a bound on the time of arrival,

$$T_{i,j} \leq T_{i-1,j-1} + \frac{\sqrt{dx^2+dy^2}}{F_{i,j}}. \tag{14}$$

Notice from the upwind condition (12) that $T_{i,j}$ is independent of $T_{i-1,j}$ if the dependency is solely from the diagonal node or another stencil. Using that upwind condition and our bound in (14), we see that there are no edge dependencies if

$$T_{i-1,j} \geq T_{i-1,j-1} + \frac{dy^2}{F_{i,j}\sqrt{dx^2+dy^2}} \geq T_{i,j} - \frac{dx^2}{F_{i,j}\sqrt{dx^2+dy^2}},$$

from which the edge causality condition follows. From the upwind condition (11), $T_{i,j}$ is not dependent on both $T_{i-1,j-1}$ and $T_{i-1,j}$ if $T_{i-1,j-1} > T_{i-1,j}$. Assuming the dependency exists, the shortest path is modelled when $T_{i-1,j-1} = T_{i-1,j}$, which gives $T_{i,j} \geq T_{i-1,j-1} + \frac{dx}{F_{i,j}}$. Investigation of the stencil including both $T_{i-1,j-1}$ and $T_{i,j-1}$ instead gives $T_{i,j} \geq T_{i-1,j-1} + \frac{dy}{F_{i,j}}$, which gives the diagonal causality condition. □

These conditions can be extended to higher order Diagonal stencils, following the approach in the derivation of a second order stencil as in Section 2.2. When using the Diagonal stencil in Fast Marching methods, the improved causality conditions apply very rarely and do thus not improve the computational efficiency. However, when using the Diagonal stencil in combination with iterative methods, these improved conditions can increase the computational efficiency significantly. According to the causality principle, any improvement of the solution in a given node may lead to improved solution values for its close neighbours. In front tracking methods, only edge-connected neighbours to an updated node needs to be updated. If there is a diagonal dependency, this will be considered at a later step in the algorithm.

Moreover, only stencils including the recently updated node should be solved for new approximations to the edge-connected nodes [9]. This *update condition* is not used in the standard Fast Marching method, and will increase the computational efficiency if applied. Not only will it save a comparison of nodal values, but also the memory handling is improved since only a smaller neighbourhood needs

to be considered. The same update condition can be used in the Fast Iterative method. Unfortunately, there is no similar update condition for the rotated stencil in the Multistencils method, since the recently passed node is edge-connected and therefore not included in the additional stencil.

4 Numerical tests

In order to make the performance comparisons fair, an identical initialisation of the boundary condition in (1) was given to all stencils by assigning parts of the domain with analytical solution values. All stencils are implemented in the same C++ Fast Marching framework, and are optimized to the same extent. The time for initialisation is excluded from the reported CPU times, which are presented as averages of five runs on a Macbook Pro with a 2.66 GHz Intel Core 2 Duo processor and 4 GB 1067 MHz DDR3 ram memory. In some examples, we have tested different levels of initialisation to show how the stencils perform with varying curvature. For all test cases we present error measurements, as estimated by the L_1, L_2 and L_∞ norms

$$\begin{aligned}
\|e\|_1 &= \frac{1}{|N|}\sum_{i\in N}|e_i|,\\
\|e\|_2 &= \left(\frac{1}{|N|}\sum_{i\in N}|e_i|^2\right)^{1/2},\\
\|e\|_\infty &= \max_{i\in N}(|e_i|),
\end{aligned}$$

where $e = T - T_a$, T and T_a is the computed and analytical solution respectively, e_i the error at node i, and N is the set of nodes to be assigned a value. When reported, the rate of convergence, p, was estimated as

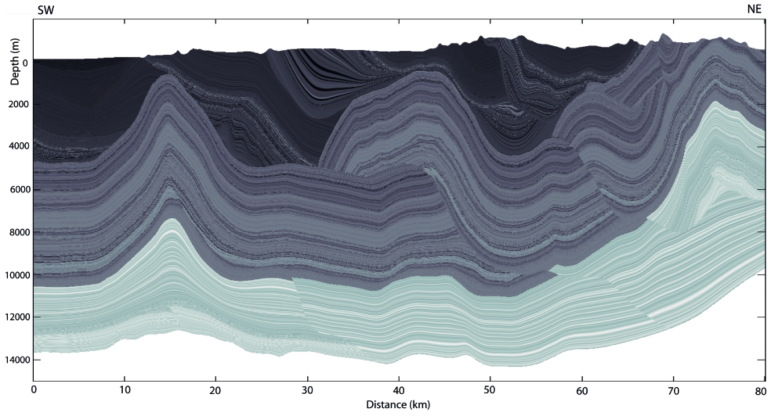
$$p = \frac{\log\left(\|e\|^{(k)}/\|e\|^{(k+1)}\right)}{\log\left(h_k/h_{k+1}\right)}, \tag{15}$$

where h_k denotes the edge length of solution k, and $\|e\|^{(k)}$ one of the corresponding norm as defined above.

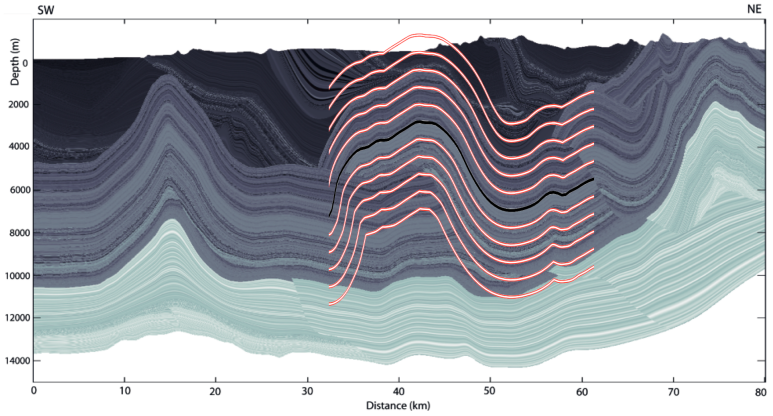
We have tested the performance of all presented stencils on an example of geological folding, and through several artificial test cases of varying complexity. In the geological example, we simulate a section of the Zagros fold belt that previously has been modelled with the CM approach [1]. In the first synthetic test, accuracy of distance computations with point sources are investigated. Thereafter, we investigate the accuracy of the stencils when there are many shocks (discontinuities) in the solution. Tests 3 and 4 have a non-constant speed function. For these two examples, the solution is first specified on parametric form T_a, and then the speed is derived by the relation $F = 1/\|\nabla T_a\|$. Convergence is investigated for the CM example, the complex speed function in test 4, and in a distance-to-point setting in test 5.

4.1 Stencil performance in the simulation of parallel folds

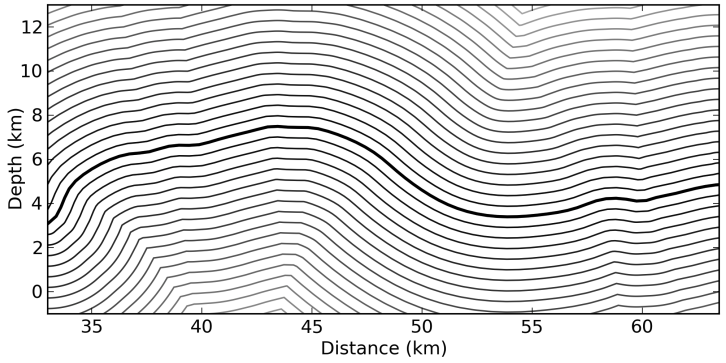
In the CM builder, a horizon is digitalized and parametrized with spline functions. In this way, the modelled horizon is smooth, assuring a continuous representation and smooth variations in dip and strike. Layers of nodes above and below the digitalized horizon are initialised using highly accurate methods based on the intersection of characteristic curves from nodes to the spline functions. However, since we do not have an analytical solution for the distance from a spline curve, we take a slightly different approach in this example. First, 100 points are sampled along the spline representation of the horizon, using uniform chord lengths. The horizon is then described by a piecewise linear curve joining



(a) CM model of a section of Zagros fold belt



(b) Same as in (a), with the simulated fold superimposed



(c) The simulated folded section

Figure 4: (a) A cross section of the investigated Zagros fold area, coloured according to acoustic impedance variations as modelled in the CM software, courtesy to [1]. (b) The same cross section with isocurves of the simulated fold superimposed in the corresponding region. (c) A simulation of the modelled folded section, visualised in equal scaling of the x- and y- axis.

	Standard G.			Multistencil			Diagonal		
dx	p_{L_1}	p_{L_2}	p_{L_∞}	p_{L_1}	p_{L_2}	p_{L_∞}	p_{L_1}	p_{L_2}	p_{L_∞}
$\frac{30.5}{120}$	0.704	0.644	0.520	0.785	0.802	0.560	0.724	0.644	0.454
$\frac{30.5}{240}$	0.771	0.705	0.532	0.799	0.775	0.688	0.787	0.687	0.550
$\frac{30.5}{480}$	0.823	0.768	0.677	0.839	0.792	0.598	0.787	0.680	0.659
$\frac{30.5}{960}$	0.849	0.775	0.696	0.865	0.794	0.648	0.831	0.743	0.729
$\frac{30.5}{1920}$	0.843	0.753	0.707	0.862	0.775	0.699	0.806	0.725	0.745
$\frac{30.5}{3840}$	0.839	0.746	0.740	0.866	0.765	0.739	0.808	0.744	0.775
$\frac{30.5}{7680}$	0.840	0.760	0.778	0.870	0.772	0.772	0.822	0.778	0.798
$\frac{30.5}{15360}$	0.829	0.769	0.801	0.856	0.776	0.803	0.805	0.784	0.820

Table 1: Convergence estimates for all first order stencils in three norms. Simulation of a folded structure, as visualized in Figure 4(c).

the points. From this representation the analytical distance field is created using the Visualization Toolkit (VTK) (`http://www.vtk.org`)[2]. All nodes with an analytical solution value less than or equal to $2\sqrt{dx^2+dy^2}$ are initially assigned their analytical distance value in all numerical runs of this example. With such an initialisation all nodes used by the stencils are on the same side of the initial horizon.

In this numerical experiment we simulate the parallel folded section in the Zagros belt positioned at distance
$x \in (33, 63.5)$ km and depth $z \in (0, 10)$ km in Figure 4(a). In Figure 4(b) we have superimposed some of the simulated layers. Figure 4(c) show a simulation of the modelled folded section, visualised with the same scaling of the x- and y- axis. In the CM software, layers above and below the fold are also mapped in the restoration process. Therefore, the computational domain is 30.5km wide and 14 km deep, where the depth interval is $(-1, 13)$ km. To get a close to uniform grid, the width is discretized with three times as many edges as the depth. For the convergence analysis we use $10 \cdot 2^i, i = 1, \ldots, 9$ edges in the depth direction. The number of unknown nodes ranges from 954 and up to 78.6 million nodes, depending on the discretization.

Tables 1 and 2 show convergence estimates for first and second order stencils, respectively. Notice that the convergence rate is rather poor for both first and second order stencils. This behaviour is to be expected. Only the closest nodes around the given horizon are initialised, and therefore the fraction of initially known nodes is decreasing as the grid expands. On the smallest and largest grids 25.5% and 0.1% of all nodes are initially assigned analytical values respectively. The convergence rates are quite similar for all stencils. Figure 5(a) shows the L_2 error given a grid size for all stencils. Among the first order stencils (thin lines), the Diagonal stencil clearly is the most accurate. The same assertion holds for the second order stencils (thick lines). Somewhat surprising, the second order Multistencil is slightly less accurate than the Standard second order stencil. This behaviour may be due to the presence of many regions with shocks in the solution, along which the added stencil in the Multistencil often creates too small traveltimes.

[2]Distances are computed using the VtkCellLocator class which uses a a uniform-level octree subdivision to accelerate distance queries.

	Standard G.			Multistencil			Diagonal		
dx	p_{L_1}	p_{L_2}	p_{L_∞}	p_{L_1}	p_{L_2}	p_{L_∞}	p_{L_1}	p_{L_2}	p_{L_∞}
$\frac{30.5}{120}$	1.092	1.115	1.006	0.985	1.112	1.174	1.057	1.016	0.300
$\frac{30.5}{240}$	1.033	1.155	0.966	1.287	1.238	1.052	0.937	0.847	0.788
$\frac{30.5}{480}$	0.865	0.929	1.224	1.090	1.122	0.779	1.230	1.147	0.920
$\frac{30.5}{960}$	1.149	1.085	0.709	1.170	1.111	0.865	1.065	0.914	1.055
$\frac{30.5}{1920}$	1.188	1.072	0.951	1.267	1.085	1.118	1.133	1.005	0.906
$\frac{30.5}{3840}$	1.146	1.007	0.961	1.258	1.089	0.902	1.077	0.971	1.000
$\frac{30.5}{7680}$	1.171	1.070	0.824	1.179	1.068	0.922	1.212	1.166	0.981
$\frac{30.5}{15360}$	1.067	0.982	1.040	1.171	1.044	1.038	1.050	0.986	1.000

Table 2: Convergence estimates for all second order stencils in three norms. Simulation of a folded structure, as visualized in Figure 4(c).

As an illustration of the efficiency of all stencils, the L_2 errors given the computational time is plotted in Figure 5(b). Again, the first order Diagonal stencil is clearly the most efficient of the first order stencils. Of the second order stencils, the Diagonal stencil is the most efficient and the Multistencil the least efficient. Plots of the L_1 and L_∞ errors are very similar. For any given grid size and stencil order, the Diagonal stencil is the fastest, followed by the Standard, and Multistencil stencils.

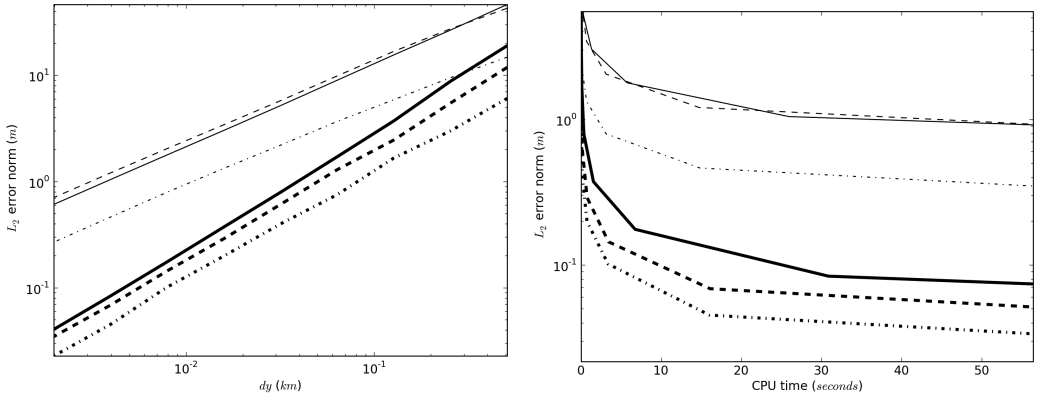
4.2 Test 1, distance to points

In this section we compute distance fields to points in two examples, tests 1A and 1B[3] First, we solve for the distance field from two points positioned at $p_1 = (35,51)$ and $p_2 = (67,51)$ in the domain $0 \leq x,y \leq 100$ discretized with 101 nodes along each axis. Isocurves of the solution are shown in Figures 6(a) and 6(b) for first and second order computations, respectively. The Diagonal stencil follows the analytical solution significantly better than both the Multistencils and Standard stencils for both first and second order methods. Table 3 reports error measures for two levels of initialisation. In the upper part only the two points p_1 and p_2 were initialised. In the lower part, all nodes for which the analytical solution is less than 4 were initially assigned the exact values. Independent of initialisation and order, the Diagonal stencil is the most accurate, followed by the Multistencil which still perform significantly better than the Standard stencil. Notice that the Multistencil iso-curves (dashed lines) are in front of the analytical solution close to the shock region. In shock regions the stencil using only diagonal nodes underestimate the traveltimes, and thus creates errors that spread throughout the domain.

We also solved for the distance to a point p on a non-uniform rectangular grid with $dx = 0.1$ and $dy = 0.2$ in test 1B. The domain is again discretize by 101 nodes along both axes, but p is now located in the center at $(x,y) = (5,10)$. From the error measures in the upper part of Table 4, we notice that the second order stencils are less accurate than the first order stencils for the Diagonal and Multistencils approaches. This is caused by errors in the second order interpolation in the diagonal directions. A value from the other side of the front, and hence not upwind, is assumed to be an upwind value. Therefore, on a strongly non-uniform discretization, a more accurate initialisation is needed for second order stencils using diagonally connected nodes. The lower part of Table 4 shows error measures correspondingly to letting nodes with analytical solution less then or equal to 0.5 be exactly initialised.

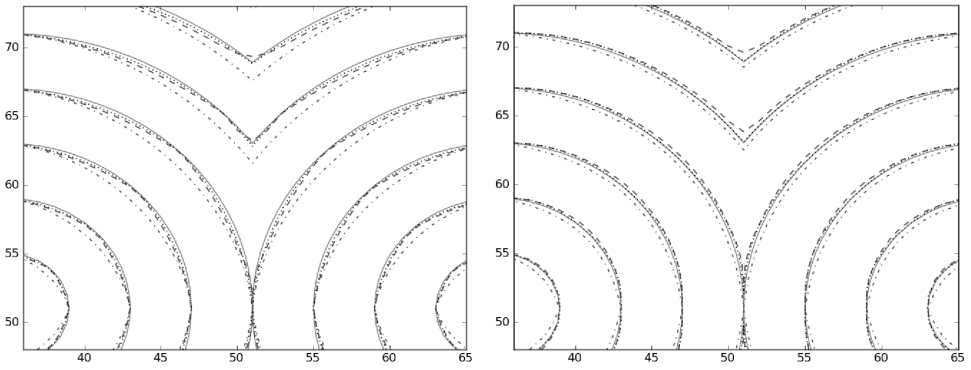
Of the first order methods the Diagonal stencil is the most accurate. The Multistencils approach has

[3]The same tests are carried out in [12].



(a) Estimated L_2 convergence for discrete width (b) Estimated L_2 convergence for CPU time

Figure 5: Error and efficiency for different stencils in the simulation of a geological fold. The number of unknown grid nodes ranges from 954 to 78.6 million nodes depending on grid resolution. Standard stencil (Dashed, '– – –'), Multistencil (Solid, '——'), Diagonal stencil (Dash-dot, '– · – ·'). Thick lines refer to second order stencil solutions. (a) L_2 error for increasing grid spacing in log-log scale. (b) $\log(L_2)$ error plotted against CPU times.



(a) First order solutions (b) Second order solutions

Figure 6: Solutions for test 1A: Analytical solution (Solid, '——'), Multistencils (Dashed '– – –'), Diagonal stencil (Dotted '····'), Standard stencil (Dash-dot '– · – ·'). Only two points are initially assigned with analytical solution values.

	First Order			Second Order		
Norms	L_1	L_2	L_∞	L_1	L_2	L_∞
Nodes with solution $= 0$ *initialised, in total 2 nodes*						
Standard	0.617	0.705	1.210	0.241	0.256	0.370
Multistencils	0.301	0.343	0.628	0.165	0.180	0.516
Diagonal	0.174	0.196	0.327	0.091	0.106	0.176
Nodes with solution ≤ 4 *initialised, in total 98 nodes*						
Standard	0.407	0.474	0.880	0.066	0.078	0.148
Multistencils	0.248	0.287	0.559	0.100	0.108	0.393
Diagonal	0.128	0.145	0.256	0.052	0.062	0.102

Table 3: Error measures for Test 1A, computed distances to two points in the domain $0 \leq x, y \leq 100$, discretized by 101 nodes in both x and y directions. Part of the domain is shown in Figure 6.

	First Order			Second Order		
Norms	L_1	L_2	L_∞	L_1	L_2	L_∞
Nodes with solution $= 0$ *initialised, in total 1 node*						
Standard	0.094	0.107	0.178	0.031	0.034	0.061
Multistencils	0.045	0.058	0.134	0.049	0.059	0.087
Diagonal	0.026	0.035	0.084	0.038	0.045	0.067
Nodes with solution ≤ 0.5 *initialised, in total 43 nodes*						
Standard	0.066	0.076	0.133	0.014	0.017	0.029
Multistencils	0.038	0.049	0.114	0.019	0.020	0.028
Diagonal	0.019	0.025	0.065	9.7e-3	0.012	0.026

Table 4: Error measures for Test 1B, distance to centred point on non-uniform grid, $dx = 0.1, dy = 0.2$, with 101 nodes in both x and y directions. The table shows errors for two different levels of initialisation.

a factor of 1.6-2 times larger errors, and the Standard stencil shows a factor of 2-3 times larger errors than the Diagonal stencil. With accurate initialisation the second order stencil accuracy order is the same. With no initialisation on the second order stencils the Standard stencil performs best, closely followed by the Diagonal and Multistencils stencils.

4.3 Test 2, shock resolution

For two different grids of the domain $0 \leq x, y \leq 10$, we solve for the distance to the domain borders and four interior points with coordinates $p_1, \ldots, p_4 = (2,8), (8,2), (3,3)$, and $(8.5, 8.5)$. As seen in Figure 7(a), there are many shocks (discontinuities in the gradient) in the analytical solution, which can be written as

$$T(x,y) = \min(x, y, 10-x, 10-y, \|(x,y)-p_i\|, i = 1, ..., 4). \tag{16}$$

Regarding the computational efficiency of this example, the Multistencils implementation requires approximately 2.3 times longer computing time than the Diagonal stencil, and the Standard stencil needs approximately 1.3 times longer time than the Diagonal stencil.

All nodes with solution values up to 0.05 have been initialised exactly, giving the error measures in Table 5. The upper part of Table 5 shows the error on a grid with 101 nodes along both axes, and the lower part shows the error on a finer grid with 501 nodes along the x axis and 601 nodes along the y axis. The Diagonal stencil is the most accurate for both first and second order approximations on both grids. The rotated stencil of the Multistencils approach underestimates the solution along shocks, as can be seen in Figure 6. This effect of underestimation appear stronger on non-uniform grids and second order stencils, and then makes the Multistencil less accurate than the Standard method.

4.4 Test 3, varying speed

This example compares the performance of the different stencils for a dynamically changing speed on a uniform grid, $dx = dy = 1$, on the domain $-28 \leq x, y \leq 28$, discretized by 57 nodes in each direction.

	First Order			Second Order		
Norms	L_1	L_2	L_∞	L_1	L_2	L_∞
$dx=\frac{1}{10}, dy=\frac{1}{10}$. *Nodes with solution* ≤ 0.05 *initialised, in total 404 nodes*						
Standard	0.025	0.042	0.112	0.012	0.018	0.037
Multistencils	0.010	0.017	0.049	7.0e-3	0.011	0.047
Diagonal	5.9e-3	0.010	0.028	3.3e-3	6.0e-3	0.017
$dx=\frac{1}{50}, dy=\frac{1}{60}$. *Nodes with solution* ≤ 0.05 *initialised, in total 6660 nodes*						
Standard	6.1e-3	0.010	0.027	1.0e-3	1.8e-3	5.4e-3
Multistencils	3.5e-3	6.2e-3	0.019	1.6e-3	2.5e-3	0.010
Diagonal	1.7e-3	2.9e-3	8.9e-3	9.9e-4	1.7e-3	4.6e-3

Table 5: Error measures for Test 2. The domain is $0 \leq x, y \leq 10$, and we solve for the distance to domain borders and four interior points (Figure 7(a)). All nodes with distance ≤ 0.05 have been initialised. The table shows error measures for two domain discretizations.

The analytical solution and the corresponding speed are given by

$$T_a = 100 - 100\cos\frac{x}{20}\cos\frac{y}{20}, \tag{17}$$

$$F_a = \frac{0.2}{\sqrt{\left(\sin\frac{x}{20}\cos\frac{y}{20}\right)^2 + \left(\sin\frac{y}{20}\cos\frac{x}{20}\right)^2}}. \tag{18}$$

With this choice of F_a, the characteristic curves are straight lines from the center point and outwards. The analytical solution is shown in Figure 7(b), and error measures for two different initialisations are presented in Table 6. Among the first order stencils, the Diagonal stencil is slightly more accurate than the Multistencils and the Standard stencil for both initialisations. The second order stencils depend more strongly on the initialisation. With only one point initialised the Multistencils approach yields the smallest error, but for a more accurate initialisation the Standard stencil is the most accurate of the second order stencils.

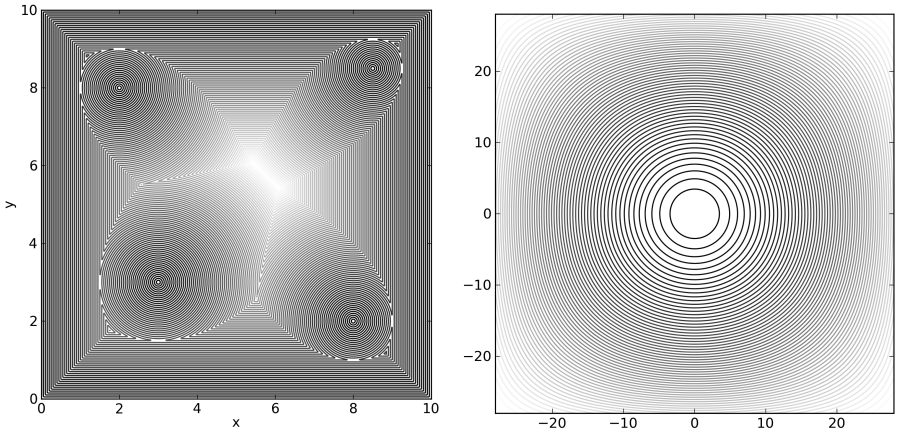
The error for the Diagonal stencil is especially large in diagonal directions. Along these directions the speed changes as $\frac{\sqrt{2}}{5}(\sin\frac{s}{10})^{-1}$, where s is the distance from the center, whereas along the axes speed is varying only as $\frac{1}{5}(\sin\frac{s}{20})^{-1}$. Since the speed is modelled piecewise constant over the 'long' diagonals where the change is the biggest, the errors due to piecewise constant speed modelling are large for the Diagonal stencil in this example.

4.5 Test 4, complex speed convergence

This test examines the convergence and accuracy when the speed changes with position. The analytical solution and its corresponding speed are defined by

$$T_a = 101y + 100\sin y\cos x,$$

$$F_a = \frac{1}{100\sqrt{\sin y^2 \sin x^2 + (1.01 + \cos x\cos y)^2}}. \tag{19}$$

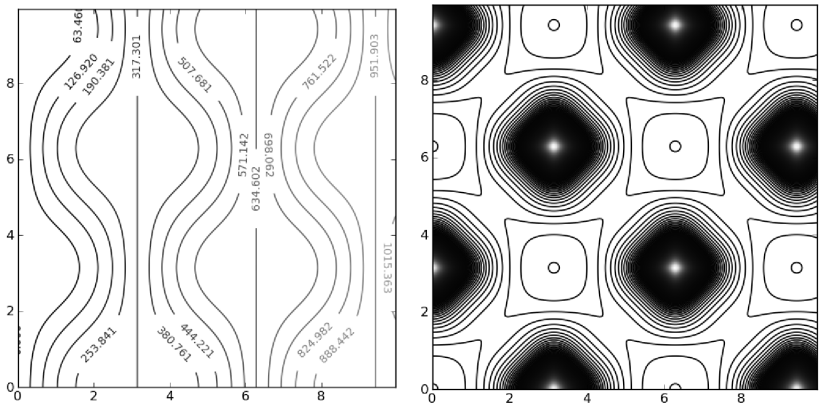


(a) Solution to test 2 (b) Solution to test 3

Figure 7: Analytical solutions to tests 2 and 3. The grey tone intensity of the isocurves is decreasing with increasing solution value.

	First Order			Second Order		
Norms	L_1	L_2	L_∞	L_1	L_2	L_∞
Node with solution $= 0$ *initialised*						
Standard	2.719	2.841	3.871	0.456	0.466	0.631
Multistencils	2.443	2.535	3.320	0.143	0.161	0.331
Diagonal	2.222	2.289	2.886	0.161	0.180	0.384
Nodes with solution ≤ 1 *initialised, in total 25 nodes*						
Standard	2.365	2.490	3.465	0.064	0.083	0.200
Multistencil	2.054	2.152	2.898	0.097	0.120	0.316
Diagonal	1.841	1.916	2.520	0.109	0.134	0.316

Table 6: Error measures for Test 3, for two levels of initialisation. First arrival time from point in the center of the domain, with speed defined by (18), on domain $-28 \leq x, y \leq 28$ discretized by 57 nodes in both x and y directions.



(a) Solution to test 4 (b) Speed to test 4

Figure 8: Analytical solution and speed to test case 4. The grey tone intensity of the isocurves is decreasing with increasing solution value.

	Standard G.			Multistencil			Diagonal		
h	p_{L_1}	p_{L_2}	p_{L_∞}	p_{L_1}	p_{L_2}	p_{L_∞}	p_{L_1}	p_{L_2}	p_{L_∞}
$\frac{1}{30}$	0.945	0.906	0.116	0.937	0.933	0.097	0.971	0.938	0.122
$\frac{1}{45}$	0.943	0.843	0.075	0.941	0.921	0.069	0.966	0.869	0.077
$\frac{1}{60}$	0.937	0.764	0.058	0.949	0.904	0.056	0.959	0.786	0.058
$\frac{1}{75}$	0.931	0.682	0.048	0.954	0.877	0.046	0.952	0.699	0.047
$\frac{1}{90}$	0.924	0.602	0.039	0.956	0.842	0.039	0.945	0.616	0.038
$\frac{1}{105}$	0.916	0.530	0.035	0.952	0.799	0.035	0.937	0.540	0.033
$\frac{1}{120}$	0.908	0.466	0.031	0.953	0.758	0.030	0.929	0.473	0.029
$\frac{1}{135}$	0.900	0.411	0.027	0.953	0.715	0.028	0.921	0.416	0.026
$\frac{1}{150}$	0.892	0.364	0.025	0.953	0.672	0.025	0.913	0.366	0.023

Table 7: Estimated order of convergence for all first order stencils, in three different norms. Test 4, with solution in Figure 8(a).

These functions are illustrated in Figures 8(a) and (b) on the domain $0 \leq x, y \leq 10$. The characteristic curves are not straight lines, but bend with the strong curvature of the solution. Only nodes with analytical value 0, that is, all nodes on the y-axis, were initialised on every run. Since the initial condition is a straight line, there is no initial discontinuity in the gradient of the solution, as is the case when points are given as initial condition. This problem was run on uniform grids for different number of nodes, starting with 151×151 and increasing with 150 nodes along each axis up to 1501. Using the solutions we estimated the convergence of all stencils, as shown in table 7 for first order stencils, and table 8 for second order stencils. Thick lines in Figure 9 are based on second order stencils and thin lines are based on first order stencils. Figure 9(a) and (b) show L_2 error for edge length and computational time respectively.

The L_∞ convergence is very low for all first order stencils, and even negative for the second order stencils due to the errors from the piecewise constant speed model. Among the first order stencils, the accuracy and efficiency of the Standard and Diagonal stencils are very similar. The first order Multistencil on the other hand is 3.5-4 times slower, and significantly less accurate than the other stencils[4]. For the second order stencils, the stencil behaviour is very different. The Diagonal second order stencils sticks out as it is significantly more accurate than any other stencil, especially on the smaller grids. At the same time the Diagonal stencil is the most efficient. The second order Multistencil behaviour is also noticeable, since it is more accurate than the second order Standard stencil. Although it is 3.2-3.6 times slower than the Standard second order stencil, the second order Multistencils approach is more efficient on larger grids.

It is not obvious why the stencils perform with such a varying performance on this example, but many effects are due to the errors from the assumption of locally constant speed. The problem itself is challenging, as the velocity dynamics compress and expand clusters of the characteristic curves at locations with high and low velocities respectively. The estimated convergence in L_1 is low for all stencils. In L_2 the convergence is even worse, and seem to be declining as the grid gets finer.

4.6 Test 5, distance to point convergence

In this test we investigate the convergence of the stencils on a simple problem with constant velocity. We solve for the distance to a point in the center of the domain $0 < x, y < 100$ for a uniform

[4] 1.35-1.87 times larger errors in L_2, and 1.96-2.05 in L_1.

	Standard G.			Multistencil			Diagonal		
h	p_{L_1}	p_{L_2}	p_{L_∞}	p_{L_1}	p_{L_2}	p_{L_∞}	p_{L_1}	p_{L_2}	p_{L_∞}
$\frac{1}{30}$	0.914	0.717	-0.108	1.544	1.125	-0.119	1.622	0.308	-0.071
$\frac{1}{45}$	0.950	0.707	-0.054	1.434	0.491	-0.078	1.124	0.070	-0.023
$\frac{1}{60}$	1.048	0.670	-0.010	1.202	0.164	-0.037	0.865	0.042	-0.009
$\frac{1}{75}$	0.778	0.405	-0.006	1.217	0.132	-0.014	0.615	0.033	-0.004
$\frac{1}{90}$	0.860	0.388	-0.006	1.141	0.081	-0.008	0.599	0.028	-0.002
$\frac{1}{105}$	0.902	0.343	-0.005	0.996	0.051	-0.006	0.289	0.023	-0.001
$\frac{1}{120}$	0.806	0.260	-0.000	0.872	0.038	-0.006	0.304	0.021	-0.001
$\frac{1}{135}$	0.940	0.258	-0.001	0.854	0.031	-0.003	0.331	0.019	-0.001
$\frac{1}{150}$	0.756	0.178	-0.002	0.741	0.025	-0.003	0.301	0.017	-0.001

Table 8: Estimated order of convergence for all second order stencils, in three different norms. Test 4, with solution in Figure 8(a).

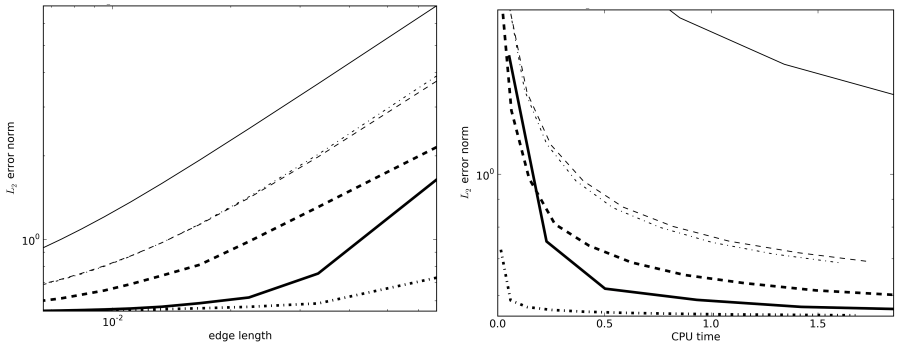
discretizations of decreasing spacing. On all grids, nodes with analytical solution less than 2 have been initialised in order to remove the otherwise increasing initial gradient discontinuity [37]. Convergence was estimated using (15) and are presented in Tables 9 and 10 for first and second order stencils, respectively.

The L_2 convergence is visualised in Figure 10(a), where the thin lines concern first order stencils and thick lines second order stencils. The corresponding L_1 and L_∞ plots are very similar. The first order convergence is close to 1 for all stencils, and the error is significantly smallest for the Diagonal stencil, followed by the Multistencils and the Standard stencils. The behaviour is very similar for second order stencils, where all stencils have an estimated rate of convergence close to 2. Again the second order Diagonal stencil has the smallest errors, followed by the Standard and Multistencils stencils.

The Diagonal stencil is the fastest for every given grid, followed by the Standard and the slower Multistencil. To illustrate the efficiency of the stencils, the L_2 error is plotted as a function of CPU time (averages of 5 runs) in Figure 10(b). First order Multistencils and Standard methods are equally efficient, but the first order Diagonal is significantly more efficient. Of the second order stencils the Diagonal stencil is the most efficient followed by the Standard stencil. The second order Multistencils is not efficient in comparison to the other stencils.

5 Discussion

Applications involving numerical solution of the eikonal equation often calls for high computational efficiency and high accuracy. In particular, this is important in seismic processing and in geological modelling, where Fast Marching methods, or similar algorithms, are used to estimate the wave's time of arrival at any spatial location in the computational domain. Motivated by these needs, we have tested three stencils in several cases that together span several of the features found in geological applications. In these tests we have compared the performance of the stencils in terms of efficiency and accuracy. Among all stencils of both first and second order, the Diagonal stencil is consistently the most efficient and accurate. The Multistencils alternative comes second in accuracy, but is by far the most computationally expensive. The Diagonal stencil is slightly faster than the Standard stencil because of the memory handling and the update condition, described in Section 3. Since such a condition does not exist for the Multistencils method, which involves a numerical solve with an additional stencil, its computational time is significantly higher.



(a) L_2 for edge length (b) L_2 for CPU time

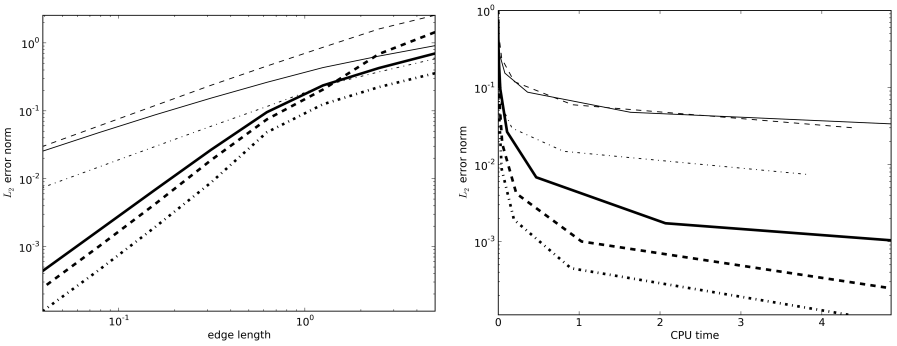
Figure 9: The L_2 error for increasing edge length in a log-log plot in Figure 9(a). $\log(L_2)$ error plotted against CPU times for Test 4 in Figure 9(b). Standard stencil (Dashed, '– – –'), Multistencil (Solid, '——'), Diagonal stencil (Dash-dot, '– · – ·'). Thick lines refer to second order stencil solutions.

	Standard G.			Multistencil			Diagonal		
h	p_{L_1}	p_{L_2}	p_{L_∞}	p_{L_1}	p_{L_2}	p_{L_∞}	p_{L_1}	p_{L_2}	p_{L_∞}
$\frac{10}{4}$	0.664	0.671	0.680	0.475	0.510	0.517	0.607	0.638	0.643
$\frac{10}{8}$	0.906	0.899	0.881	0.550	0.563	0.583	0.675	0.684	0.694
$\frac{10}{16}$	0.922	0.915	0.904	0.725	0.721	0.715	1.007	1.011	1.003
$\frac{10}{32}$	0.948	0.945	0.940	0.778	0.776	0.781	0.969	0.969	0.963
$\frac{10}{64}$	0.994	0.994	0.991	0.824	0.821	0.820	0.993	0.992	0.988
$\frac{10}{128}$	0.996	0.996	0.994	0.869	0.867	0.866	1.000	1.000	0.996
$\frac{10}{256}$	0.997	0.997	0.995	0.903	0.903	0.901	0.999	1.000	0.996

Table 9: Estimated order of convergence for all first order stencils, in three different norms. Test 5, distance computation to a point in the center of the domain $0 \leq x, y \leq 100$ on a uniform discretization.

	Standard G.			Multistencil			Diagonal		
h	p_{L_1}	p_{L_2}	p_{L_∞}	p_{L_1}	p_{L_2}	p_{L_∞}	p_{L_1}	p_{L_2}	p_{L_∞}
$\frac{10}{4}$	1.044	1.066	1.000	0.640	0.711	0.874	0.622	0.676	0.773
$\frac{10}{8}$	1.945	1.742	1.465	0.816	0.852	0.943	0.819	0.843	0.933
$\frac{10}{16}$	1.417	1.457	1.491	1.239	1.308	1.326	1.262	1.370	1.631
$\frac{10}{32}$	1.958	2.003	2.060	1.860	1.848	1.731	2.518	2.465	2.367
$\frac{10}{64}$	2.134	2.151	2.162	1.991	1.958	1.797	2.178	2.183	2.081
$\frac{10}{128}$	2.093	2.069	2.028	1.994	1.982	1.925	2.088	2.095	2.101
$\frac{10}{256}$	2.040	2.018	1.879	1.984	1.979	1.885	2.009	2.004	1.782

Table 10: Estimated order of convergence for all second order stencils, in three different norms. Test 5, distance computation to a point in the center of the domain $0 \leq x, y \leq 100$ on a uniform discretization.



(a) L_2 for grid spacing (b) L_2 for CPU time

Figure 10: Error and efficiency illustrations for Test 5. (a) L_2 error for decreasing grid spacing in a log-log plot. (b) $\log(L_2)$ error plotted against CPU times. Standard stencil (Dashed, '– – –'), Multistencil (Solid, '——'), Diagonal stencil (Dash-dot, '– · – ·'). Thick lines refer to second order stencil solutions.

It is well known that the Standard stencil generates large errors especially in diagonal directions. Therefore it makes sense to include diagonal nodes in the stencil shape. Inherently, all discussed stencil shapes assume a linear interpolation between nodes. The Diagonal shape uses nodes close to each other, whereas the Standard and Multistencil shapes use more distant nodes. Thus, there is an increased possibility of errors in the linear interpolation assumption. Furthermore, it is important that the nodes used in the update step are upwind of the node to be updated. In a front propagation context the point of the wave that pass through the updated node must originate from inside the linear interpolated wave segment of the used nodes. That is, the shortest path (characteristic curve) cuts through the convex hull of the used nodes before reaching the node to be updated. For the eikonal equation, efficient upwind conditions can be formulated for all considered stencils. The Diagonal shape uses nodes close together in the upwind direction to get a more accurate estimate of the arriving wave front, and thus a better solution estimate. Although we have only tested different shapes for the isotropic eikonal equation, we expect the findings to hold also for more general front propagations. This is indicated in the numerical experiments performed in [10].

Stencils using diagonal nodes are not necessarily more accurate when the speed is changing. This is due to longer "shortest" paths being considered in the stencils where the speed is constant. The longest path is $\sqrt{dx^2+dy^2}$ instead of $\max(dx,dy)$. However, this seems only to affect the stencils using diagonal nodes in extreme cases when the velocity changes the most in diagonal directions. For smooth variations of the velocity, the Diagonal stencil outperforms the other stencils, especially so for second order stencils. A reason for this is that only nodes from a small neighbourhood are used in the Diagonal second order update. Other stencils use nodes more distant from each other, and thus further violates the underlying assumption of a constant underlying velocity. For non-smooth speeds, all the presented stencils show slow convergence. Instead, other modelling approaches should be considered. For instance, interpolation of the speed along linear shortest paths have been shown to significantly increase accuracy, unfortunately to a high computational price [4].

Second order stencils use two nodes in the same direction to get a more accurate gradient estimate. If the more distant node is not upwind of the closer node, the provided solution estimate might be too small, and generate errors that spread through the domain. A condition for the more distant node to be upwind is that its solution value is smaller than at the closer node. However, when the wave front is very curved this condition is not sufficient. On very non-uniform grids, second order stencils using diagonal nodes seem more sensitive to this condition than the Standard stencil. The initial waveform often has a high curvature, and care should therefore be taken when initialising the algorithms. Moreover,

the accuracy of the initialisation procedure is very important. For proper convergence analysis, all nodes up to a fixed value should be given analytical values. Otherwise, the initial discontinuity in the gradient will increase as the grid gets finer, which reduces the rate of convergence of the algorithms. Although second order stencils use second order upwind finite difference estimates, they do not in general provide second order solution estimates. However, second order stencils provide significantly more accurate solution estimates at almost no increase in the computational cost.

6 Conclusion

We have derived the Standard, Multistencils, and Diagonal stencils for the isotropic eikonal equation. Using the approach of modelling the wave front directly, we have obtained valuable insight regarding causality and upwind conditions. Moreover, by interpreting second order upwind schemes as forward interpolations, we have introduced a method for generalising first order upwind stencils of different shapes to second order. This method was used to derive the second order variant of the Diagonal stencil.

All stencils were compared in an extensive range of tests, indicating that the Diagonal stencil is the optimal choice. The Standard stencil also performs consistently well. It is most often slightly less accurate than the Multistencil, but is also considerably faster. Although the Multistencil approach is rather accurate, its performance changes with problem setting. Moreover, the computational efforts are increased with an amount that might challenge the gain in accuracy compared to the Standard stencil. The Diagonal stencil is always the faster choice, and moreover provides a more accurate solution. These findings are consistent for both first and second order. Most reports of eikonal solvers use the Standard stencil. Our investigation indicates that Diagonal stencils are attractive for demanding applications, such as in geological modelling.

Acknowledgements We thank Tangui Morvan for the idea, and implementation of, the analytical solution for the Zagros fold simulation using VTK. The presented work was funded by Statoil through the Akademia program, and the Research Council of Norway under grant 202101/I40. The work has been conducted at Kalkulo AS, a subsidiary of Simula Research Laboratory.

Bibliography

1. Alaei, B. and S. A. Petersen (2007). Geological modelling and finite difference forward realization of a regional section from the Zagros fold-and-thrust belt. *Petroleum Geoscience 13*(3), 241–251.

2. Alkhalifah, T. (2001). Implementing the fast marching eikonal solver: Spherical versus cartesian coordinates. *Geophysical Prospecting 70*(2), 245–178.

3. Antiga, L. and D. A. Steinman (2010). VMTK: the Vascular Modeling Toolkit.

4. Appia, V. and A. Yezzi (2010). Fully isotropic fast marching methods on cartesian grids. In K. Daniilidis, P. Maragos, and N. Paragios (Eds.), *Computer Vision – ECCV 2010*, Lecture Notes in Computer Science, pp. 73–85. Springer Berlin/Heidelberg.

5. Bak, S., J. McLaughlin, and D. Renzi (2010, September). Some improvements for the fast sweeping method. *SIAM Journal on Scientific Computing 32*(5), 2853–2874.

6. Bancroft, J. and X. Du (2005). Circular Wavefront Assumptions for Gridded Traveltime Computations. *CREWES Report 17*, 1–14.

7. Berre, I., K. H. Karlsen, K.-A. Lie, and J. R. Natvig (2005, November). Fast computation of arrival times in heterogeneous media. *Computational Geosciences 9*(4), 179–201.

8. Bruaset, A. M. (2010). *Simula Research Laboratory – by thinking constantly about it*, Chapter Turning Rocks into Knowledge, pp. 553–600. Springer.

9. Danielsson, P.-E. and Q. Lin (2003). A modified fast marching method. In *SCIA'03: Proceedings of the 13th Scandinavian conference on Image analysis*, Berlin, Heidelberg, pp. 1154–1161. Springer-Verlag.

10. Gillberg, T. (2011). A semi-ordered fast iterative method (SOFI) for monotone front propagation in simulations of geological folding. In *MODSIM2011, 19th International Congress on Modelling and Simulation*, pp. 641–647.

11. Gremaud, P. A. and C. M. Kuster (2006, December). Computational study of fast methods for the eikonal equation. *SIAM Journal on Scientific Computing 27*(6), 1803–1816.

12. Hassouna, M. S. and A. A. Farag (2007, September). Multistencils fast marching methods: A highly accurate solution to the eikonal equation on cartesian domains. *IEEE Transactions on Pattern Analysis and Machine Intelligence 29*(9), 1563–1574.

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

14. Huang, J.-W. and G. Bellefleur (2012). Joint transmission and reflection traveltime tomography using the fast sweeping method and the adjoint-state technique. *Geophysical Journal International 188*(2), 570–582.

15. Hysing, S.-R. and S. Turek (2005). The eikonal equation: numerical efficiency vs. algorithmic complexity on quadrilateral grids. In *Proceedings of ALGORITMY*.

16. Jeong, W.-K. and R. T. Whitaker (2007). A fast iterative method for a class of Hamilton-Jacobi equations on parallel systems. Technical report, University of Utah.

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

18. Kadlec, B. and G. Dorn (2010). Leveraging graphics processing units (GPUs) for real-time seismic interpretation. *The Leading Edge 29*(1), 60–66.

19. Kim, S. (2001). An $O(N)$ level set method for eikonal equations. *SIAM Journal on Scientific Computing 22*(6), 2178–2193.

20. Kim, S. (2002, July). 3-D eikonal solvers: First-arrival traveltimes. *Geophysics 67*(4), 1225–1231.

21. Kimmel, R. and J. A. Sethian (2001). Optimal algorithm for shape from shading and path planning. *Journal of Mathematical Imaging and Vision 14*, 237–244.

22. Lambaré, G., P. S. Lucio, and A. Hanyga (1996). Two-dimensional multivalued traveltime and amplitude maps by uniform sampling of a ray field. *Geophysical Journal International 125*(2), 584–598.

23. Li, H., A. Elmoataz, J. Fadili, and S. Ruan (2004, September). Dual front evolution model and its application in medical imaging. In *MICCAI 2004*, Saint Malo, France.

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

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

26. Perrin, M., B. Zhu, J.-F. Rainaud, and S. Schneider (2005). Knowledge-driven applications for geological modeling. *Journal of Petroleum Science and Engineering 47*(1–2), 89 – 104. Intelligent Computing in Petroleum Engineering.

27. Petersen, S. A. (1999). Compound modelling, a geological approach to the construction of shared earth models. *EAGE 61th Conference & Exhibition, Extended Abstracts*.

28. Petersen, S. A. (2004). Optimization strategy for shared earth modeling. *EAGE 66th Conference & Exhibition, Extended Abstracts*.

29. Petersen, S. A. and Ø. Hjelle (2008). Earth recursion, an important component in shared earth model builders. *EAGE 70th Conference & Exhibition, Extended Abstracts*.

30. Petersen, S. A., Ø. Hjelle, and S. L. Jensen (2007). Earth modelling using distance fields derived by fast marching. *EAGE 69th Conference & Exhibition, Extended Abstracts*.

31. Podvin, P. and I. Lecomte (1991). Finite difference computation of traveltimes in very contrasted velocity models: a massively parallel approach and its associated tools. *Geophysical Journal International 105*, 271–284.

32. Popovici, A. M. and J. A. Sethian (1999). 3-D traveltime computation using the fast marching method. *Geophysics 64*(2), 516–523.

33. Popovici, A. M. and J. A. Sethian (2002). 3-D imaging using higher order fast marching traveltimes. *Geophysics 67*(604, Issue 2), 604 – 609.

34. Qin, F., Y. Luo, K. B. Olsen, W. Cai, and G. T. Schuster (1992, March). Finite-difference solution of the eikonal equation along expanding wavefronts. *Geophysics 57*(3), 478–487.

35. Rawlinson, N., J. Hauser, and M. Sambridge (2009). Seismic ray tracing and wavefront tracking in laterally heterogeneous media. *Advances in geophysics 49*, 203–267.

36. Rawlinson, N. and M. Sambridge (2004). Multiple reflection and transmission phases in complex layered media using a multistage fast marching method. *Geophysics 69*(5), 1338–1350.

37. Rickett, J. and S. Fomel (1999, April). A second-order fast marching eikonal solver. In *Stanford Exploration Project Report*, Volume 100, pp. 287–292. SEP.

38. Sethian, J. A. (1999). *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science*. Cambridge University Press.

39. Sethian, J. A. and A. Vladimirsky (2001). Ordered upwind methods for static Hamilton-Jacobi equations. *Proceedings of the National Academy of Sciences of the United States of America 98*(20), 11069–11074.

40. Tsitsiklis, J. N. (1995). Efficient algorithms for globally optimal trajectories. *IEEE Transactions on Automatic Control 40*(9), 1528–1538.

41. Vidale, J. E. (1988). Finite-difference calculation of travel times. *Bulletin of the Seismological Society of America 78*(6), 2062–2076.

42. Weber, O., Y. S. Devir, A. M. Bronstein, M. M. Bronstein, and R. Kimmel (2008). Parallel algorithms for approximation of distance maps on parametric surfaces. *ACM Transactions on Graphics (TOG) 27*(4), 104:1–104:16.

43. Xu, S.-G., Y.-X. Zhang, and J.-H. Yong (2010). A fast sweeping method for computing geodesics on triangular manifolds. *IEEE Trans. Pattern Anal. Mach. Intell. 32*(2), 231–241.

44. Zhang, Y.-T., S. Chen, F. Li, H. Zhao, and C.-W. Shu (2011). Uniformly accurate discontinuous galerkin fast sweeping methods for eikonal equations. *SIAM Journal on Scientific Computing 33*(4), 1873–1896.

45. Zhang, Y.-T., H.-K. Zhao, and J. Qian (2006). High order fast sweeping methods for static Hamilton-Jacobi equations. *Journal of Scientific Computing 29*(1), 25–56.

46. Zhao, H.-K. (2004, May). A fast sweeping method for eikonal equations. *Mathematics of Computation 74*(250), 603–627.

Paper II:
A semi-ordered fast iterative method (SOFI) for monotone front propagation in simulations of geological folding

A semi-ordered fast iterative method (SOFI) for monotone front propagation in simulations of geological folding

Tor Gillberg [1,2]

[1] Computational Geosciences, Simula Research Laboratory,
P. O. Box 134, N-1325 Lysaker, Norway

[2] Kalkulo AS,
P. O. Box 134, N-1325 Lysaker, Norway

Proceedings of *MODSIM 2011, 19th International Congress on Modelling and Simulation*, pages 641-647.

Abstract:

This paper present a novel algorithm for monotone front propagation of anisotropic nature. In several examples the new algorithm is shown to be fast and able to solve a general class of front propagation problems. The algorithm is inspired by Huygens' principle in that the front is described using a list of nodes that are used as source points to evolve the front. Nodes affected by the source points are either directly used as source points or temporarily paused, depending on their solution value and the average solution value of all source points. This feature makes the algorithm semi-ordered. Still, nodes may be used as source points several times, making the algorithm iterative of nature. Together, these features create the Semi-Ordered Fast Iterative (SOFI) method.

Unlike other iterative algorithms the performance does not depend strongly on the domain geometry or variations in front velocity. Instead, the performance seems closer to that of the more stable Ordered Upwind Methods. We compare the computational time between the SOFI and Fast Marching method for an increasing grid on two isotropic examples. The computational time of the SOFI method is shorter than that of the Fast Marching method, especially on large grids. Ordered Upwind Methods have a computational scaling of $O(N \log N)$, where N is the total number of unknown nodes. The $\log N$ factor stems from the sorting needed for the front propagation. The SOFI method needs no sorting, and our numerical experiments indicate that it is of order $O(N)$.

On isotropic examples the SOFI method solutions are identical to those from the Fast Marching method assuming the same stencil is used in both methods. On problems with anisotropy the solutions are identical to those from the Fast Sweeping method when the same stencils are used. The SOFI method has many similarities with two recently introduced iterative methods, the Fast Iterative method and the Two Queue method. The Fast Iterative method lacks the semi-ordering, and its performance is therefore very problem dependent. The Two Queue method also pauses nodes to get a partially ordered method, but is only applicable to isotropic problem formulations.

Stencils of different forms can be used with minor modifications of the algorithm. We present examples where the stencil uses only edge connected nodes, and also when diagonal nodes are included in the stencil. The SOFI method can use any consistent local wave approximation (stencil), and may therefore keep the constant velocity assumption to a small area unlike the Ordered Upwind Methods which often assume that the velocity profile is constant in a larger neighbourhood. In geoscience, the simulation of an expanding front is used for the modelling of structural folds. Modelling of geological folding is a key component of the shared earth model the *Compound Earth Simulator*, developed by the oil and gas company Statoil. Non-parallel folds are modelled using anisotropic front propagation, where the velocity of the front depends on the direction the front is moving. Three different classes of folds are illustrated in our example section, all created using the SOFI method.

1 Introduction

The simulation of an expanding front is needed in many scientific applications. In the Earth sciences fast solutions are used in forward modelling of seismic data (Rawlinson and Sambridge [13]), to describe complex folding structures in geological systems (Hjelle and Petersen [7]), and in reservoir simulations (Berre et al. [3]).There are many applications in other fields discussed by Sethian [14], including grid generation, optimal path planning, and computer visualisation applications.

Statoil is currently developing a shared earth model; the *Compound Earth Simulator* or for short Compound. The Compound framework models an earth segment by combining drill measurements and seismic data with human intuition. A user can restore faults and folded structures interactively, and furthermore simulate the geological evolution by placing geological events and processes along a timeline. A key component in Compound is the modelling of geological folds as described in Hjelle and Petersen [7]. To obtain a user-friendly and interactive application the simulations must be fast and accurate. Both speed and accuracy can be increased by using an alternative stencil formulation that include nodes diagonal to the point being updated (Gillberg et al. [5]).

2 Background

In this section we briefly discuss a mathematical framework for monotone front propagation, and two concepts of importance to our algorithm design.

Front propagation

The expanding front is described by its (first) time of arrival, T, to all points in a domain Ω from the start position Γ. A general mathematical framework for the time-of-arrival is given by static Hamilton-Jacobi equations of the form

$$H(x,\nabla T) = 0, \;\; T(x) = 0 \;\forall x \in \Gamma, \tag{1}$$

where H is convex in ∇T. The solution $T(x)$ can be thought of as the distance from x to Γ, as measured with a metric defined by H. Solutions to the Eikonal equation $H(x,\nabla T) = \nabla T \cdot \nabla T - \frac{1}{F(x)^2}$ with $F \equiv 1$, is the minimal Euclidian distance from x to Γ.

Causality

What lies ahead of a moving front does not affect the past movements of the front. This property is known as the causality principle. For monotone front propagation causality states that smaller values do not depend on larger ones. Causality implies that the solution should be constructed in an increasing order, an approach heavily exploited by several algorithms, know as front tracking methods.

On a discrete setting the causality principle cannot be directly employed, since larger valued nodes may affect the solution of smaller ones. This is due to the way the front is modelled in the update step, that is the numerical stencils. All stencils use some sort of interpolation of values between nodes. However, for an upwind-stencil we can formulate the following discrete causality principle. The value at node D, T_D, is to be approximated using solution values to nodes of the set $\{U_1,\ldots,U_n\}$.

Observation 1. *The value at D might depend on the values of* $\{U_1,\ldots,U_n\}$ *only if* $\min_{\{U_1,\ldots,U_n\}} T_{U_i} < T_D$.

If $\min T_{U_i i=1,\ldots,n} \geq T_D$ then node D is upwind (behind) all of $U_1,\ldots,U_n$, and the front will first pass node D and later nodes $U_1,\ldots,U_n$. If $U_i < T_D$ for some i, then node D is downwind of U_i, and the time the front reaches D may depend on the time the front first reached U_i, and therefore $\{U_1,\ldots,U_n\}$.

Observation 1 is a weak, but very general, discrete formulation of the causality principle that is valid for any upwind finite difference stencil for front propagation. With a given stencil and propagation formulation it is often possible to come up with a stricter discrete causality formulation, as done for the Eikonal equation in Gillberg et al. [5].

Huygens' principle

Huygens' principle is formulated as follows by [1]:

All points on a wavefront serve as point sources of secondary wavelets. After a short time the new position of the wavefront will be that of the surface tangent to those secondary wavelets.

Instead of following the entire front continuously, one can look at the front as described by a set of source points. The combined front (envelope) of all source points gives the new front position. In short, the method presented in section 4 is a discrete version of Huygens' principle, which traces the front using the mean solution value of the discrete source points.

3 Front propagation algorithms

The algorithms that exploit the causality principle the most are the front tracking methods, such as the Fast Marching and Expanding Wavefront Methods, presented by Sethian [14] and Qin et al. [11]. Front tracking methods approximate the front, and use the point on the front with minimal value as a source point to further evolve the front, before considering the point as passed by the front. Since a node is only passed one time, these algorithms are called one-pass methods. In order to know which node is to be passed next, an ordered data structure is needed. Therefore, parallel implementations of these algorithms are difficult to achieve. The method can be made faster by passing a set of nodes close to the wave front simultaneously as suggested by Kim [9]. Extensions to anisotropic propagation are known as Ordered Upwind Methods. These methods are complicated, some must be simplified to be implemented, and need prior knowledge of the degree of anisotropy in the problem, see for instance Alton [1], Cristiani [4], Sethian and Vladimirsky [15]. The Ordered Upwind methods use a large neighbourhood when updating a node. By doing so, an underlying assumption is that the velocity is constant in the larger neighbourhood, which is not the case for problems with local anisotropy.

Another approach is to sweep the front in a set of predefined directions with Gauss-Seidel iterations, see Qian et al. [10], or by assuming the front will have a spherical-like shape as done by Vidale [16]. These iterative methods are sensitive to domain geometry and variations in the velocity, and are therefore often slower than front tracking methods. Complex domain geometries and velocity variations may cause the solution dependencies, the characteristic curves, to twist and bend and many iterations are often needed. Recently, two iterative algorithms that make partial use of the causality principle have been presented, the Fast Iterative method by Jeong and Whitaker [8], and the Single/Two Queue methods by Bak et al. [2]. The Fast Iterative and Single Queue methods have an active list (queue) of nodes to expand the wave everywhere simultaneously, and their performance is highly problem dependent. The Two Queue method enforces a better use of causality, and is therefore less problem dependent than the Fast Iterative and Single Queue methods. The Queue methods need fewer operations than the Fast Iterative method, but they are only capable to solve isotropic front propagation problems.

4 The semi-ordered fast iterative method

Let Ω_D denote the set of nodes on which we wish to compute the time of arrival, and assume that the initial distance is known at the nodes $\Gamma_D \subset \Omega_D$. Initially, we assume that the front does not reach

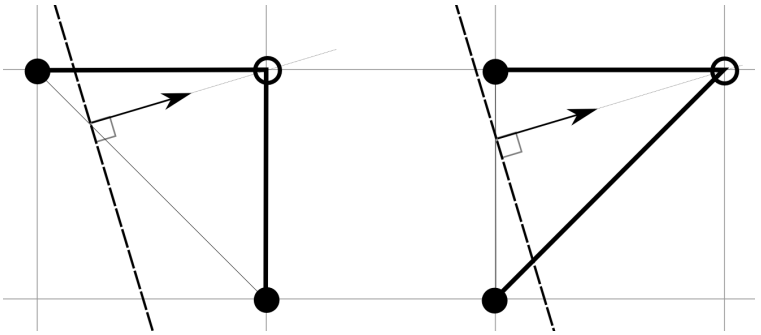


Figure 1: Two stencil forms, where the arrow points to the node being updated. The left stencil shows a edge stencil, and the right a diagonal stencil.

any nodes that are not initialised, $T(x) = \infty \; \forall x \in \Omega_D \setminus \Gamma_D$. The current front is described by a list, `aL`, containing source points (active nodes). In the initialisation step, all initialised nodes are added to `aL`. We also have a list of nodes who are ahead of the source points, `pL`, that initially is empty but later will contain paused nodes, that is nodes that later will be used as source points.

As in Huygens' interpretation of a moving front, all nodes in `aL` are used as source points to evolve the front. Using observation 1, close nodes potentially downwind of a source point are updated using stencils where the source point is included. In the observation, $U_1, \ldots, U_n$ are all nodes in the stencil, and D is the close node. The definition of close nodes depends on the stencil form. For example if the stencil only uses edge connected nodes (left stencil of figure 1) only edge connected neighbours to the source point need to be updated. If the stencil uses both edge and diagonal neighbours on a regular grid (right stencil form in 1), the diagonal nodes of the source point may also need to be updated. These corresponding stencil forms will be referred to as a diagonal and edge stencil respectively. The diagonal stencil is the more accurate since it makes use of a more local wave approximation, see Gillberg et al. [5] for details.

We construct the solution in a semi-ordered fashion using a parameter av. Assume that node x_n receives a new solution value that is smaller than the old value, $t < T(x_n)$. If in addition $t \leq av$ then x_n is added to the end of `aL`, and used as a source point. If instead $t > av$, we postpone its function as a source by adding x_n to `pL`. The approach of using two lists was first suggested by Glover et al. [6]. Let m_k be the average solution value of all nodes added to `pL` during iteration $k-1$ of the `aL` list. By choosing $av = m_k$ we enforce causality with no prior information of the problem. The method may benefit by relaxing the semi-ordering by choosing $av = 1.3m_k - 0.3m_{k-1}$[5]. The relaxation is beneficial if an edge stencil is used but not noticeably when a diagonal stencil is used. When there are no source points left, no nodes in Ω_D can get a lower arrival time. Any upwind stencil with a proper upwind condition can be applied if the source point neighbours are correctly defined. Pseudo code for the full algorithm is given below.

When both lists are empty, the algorithm has assured that no nodes in Ω_D can possibly get a lower solution value, since all dependencies have been exploited. Any upwind stencil with a proper upwind condition can be used, as long as the source point neighbours are correctly defined.

5 Numerical verification

If the stencil uses diagonally connected nodes, nodes diagonal of the source point should be updated. However, when not mentioned otherwise, the semi-ordering of SOFI assures a correct solution with

[5] Approximately 50% of the nodes are activated with no relaxation. [2] suggest that 65–75% are optimal.

Algorithm 4.1: SOFI()

Initialise T, aL, empty pL, and set $av = 0$.
while aL is not empty
do
- **for each** $x \in$ aL
- **do**
  - Remove x from aL
  - **for each** x_n node, that is possibly downwind of x
  - **do**
    - $t_{new} \leftarrow$ {New estimate of x_n value, using stencils including node x}
    - **if** $T(x_n) > t_{new} > av$
      then Add x_n to pL
    - **else if** $T(x_n) > t_{new}$ and $t_{new} \leq av$
      then Add x_n to aL
    - $T(x_n) \leftarrow \min(T(x_n), t_{new})$
- **if** aL is empty
  then
  - SWITCH(aL, pL)
  - Update av

only edge connected updates on the examples in this section. In case of highly irregular speed functions, or strong anisotropy, the diagonal update should be included to get solutions with the smallest errors.

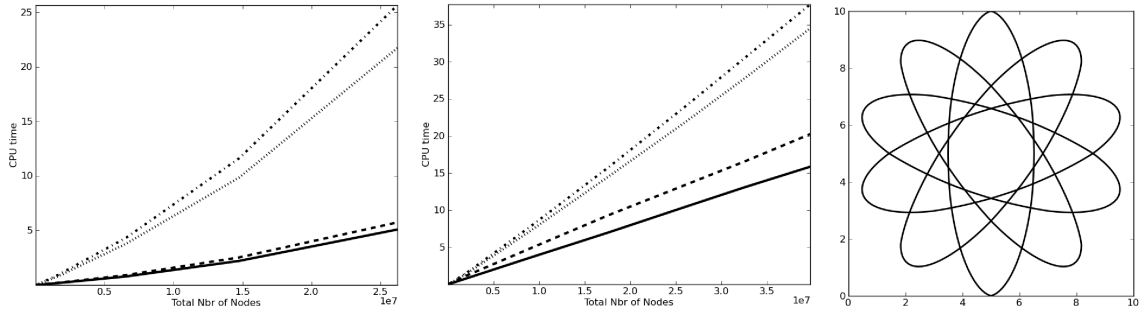
5.1 Computational order and the Hamilton-Jacobi-Bellman equation

In order to illustrate the computational order, graphs in figures 2(a) and 2(b) show CPU times for an increasing number of nodes for both the Fast Marching and SOFI methods applied to two isotropic problems. Both methods has been implemented in C++, compiled with O3 optimization, and uses identical stencil formulations. Solid (dashed) lines are CPU times for the SOFI method with a diagonal (edge) stencil, and dotted (dash-dotted) lines are CPU times for the Fast Marching method with a diagonal (edge) stencil. All computational times are averages of 5 runs on a MacBookPro with a 2.66 GHz Intel Core 2 Duo processor and 2×2 GB 1067 MHz DDR3 ram memory. The two equations which are solved are

$$\|\nabla T\| = 1, \qquad T(50,50) = 0, \qquad 0 \leq x,y \leq 100 \quad (2)$$

$$\frac{\|\nabla T\|}{100\sqrt{\sin y^2 \sin x^2 + (1.01 + \cos x \cos y)^2}} = 1, \qquad T(x,0) = 0, \qquad 0 \leq x,y \leq 10. \quad (3)$$

The characteristic curves of (2) are straight lines, but they are curved for (3) which indicates that iterative methods perform badly. For a given stencil the SOFI method is 2.4-4.8 times faster depending on grid sizes for (2). For the more complex problem of (3), the computations of the velocity are very time consuming, and the SOFI method is therefore only 1.1-2.2 times faster. Notice that the diagonal stencil is faster for both methods, supported by [5]. The Fast Marching method has a computational scaling of $O(N \log N)$, visible especially on the simple distance computation. There is no sorting in the



(a) CPU time, distance to centered point (b) CPU time, complex isotropic speed (c) Five 5-isocurves to (4)

Figure 2: (a) and (b) show the computational time in seconds for the total number of nodes for (2) and (3). Solid (dashed) lines are CPU times for the SOFI method with a diagonal (edge) stencil, dotted (dash-dot) lines are CPU times for the Fast Marching method with a diagonal (edge) stencil. Figure (c) shows the 5-isocurve for five solutions with different anisotropic directions of (4).

SOFI method, and every source point has a constant number of neighbour nodes. Together with figures 2(a) and 2(b), these observations implies a computational scaling of $O(N)$ for the SOFI method. The SOFI method solution is identical to the Fast Marching solution for both equations.

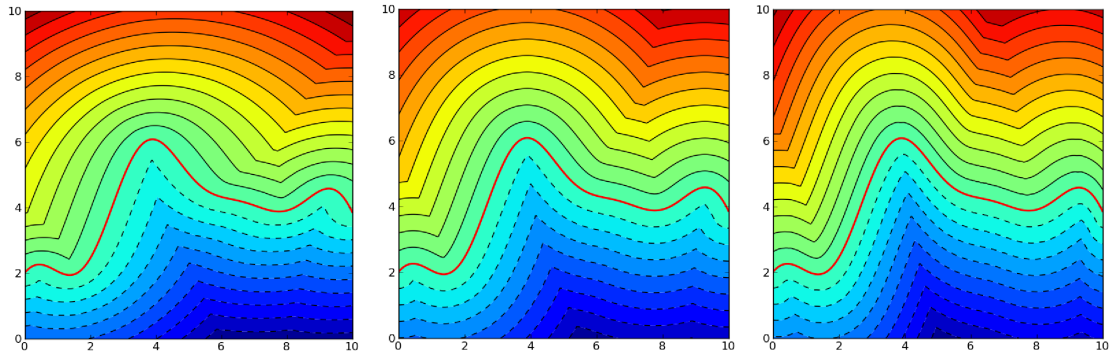
We counted the average number of stencil solves per node for these isotropic problems on a grid of 560×560 nodes. For equation (2) the SOFI method needs 1.996 updates for both stencil types, precisely the same number as the Fast Marching method. The Two Queue method need 1.626, and the Single Queue method only 1, update per node (Bak et al. [2]). The performance for problem (3) is very different. Here the SOFI method with a diagonal stencil use 2.047, and with an edge stencil 2.906 stencil solves. The Fast Marching method need on average 1.998 updates for both stencil types. The Single Queue method need 186.863, and the Two Queue method with dynamic queue cutoff 2.312 updates per node. A significant drawback for the Two Queue methods is that the average speed is needed for the cutoff implementation. The average speed can be both costly and difficult to compute.

Our first anisotropic example is of Hamilton Jacobi Bellman type, describing the distance from a point in a tilted plane, $z = \mathbf{c}_1 x + \mathbf{c}_2 y$. The problem is formulated by

$$\min_{\mathbf{u}\in B^2} \frac{(\nabla T(x)\cdot -\mathbf{u})}{(1+(\mathbf{c}\cdot\mathbf{u})^2)^{1/2}} = 1, \;\; 0 \le x,y \le 1, \;\; T(0.5,0.5) = 0, \tag{4}$$

where B^2 is the set of all unit vectors. On a discrete setting the causality principle cannot be directly employed, since larger valued nodes may affect the solution of smaller ones. Therefore (4) cannot be solved by a Fast Marching approach ([4, 15]).

Figure 2(c) shows the 5-isocurve for solutions to (4) with $\mathbf{c} = \sqrt{10}(\sin\frac{i\pi}{5}, \cos\frac{i\pi}{5})$ for $i = 1,2,3,4,5$, created on a grid of 101×101 nodes. The used stencils are of diagonal form. In the update step the stencil including the smaller diagonal node is first solved, and thereafter the stencil including the larger diagonal node is considered. The average number of updates per node if only edge neighbours are updated are 5.206 per node. If also nodes diagonal from the source point are included the average number of operations increases to 7.415. The degree of anisotropy, as described in Sethian and Vladimirsky [15], for this problem is $\sqrt{1+|\mathbf{c}|^2} = \sqrt{11}$. If we increase the degree of anisotropy to $\sqrt{21}$, diagonal nodes from the source should be updated to get a solution with the smallest errors. The number of operations per node then increases to 10.952. The solution of these examples are identical to that of the Fast Sweeping method if the same stencils are used, and numerical convergence estimates for similar anisotropic equations are presented in Qian et al. [10].



(a) Fold of class 1A (b) Fold of class 1B (parallel) (c) Fold of class 1C

Figure 3: Three modelled folds of different classes, from left class 1A, 1B, 1C after Ramsay's classification system, as explained in [7]. The initial horizon, marked with red, and the anisotropic direction, $\mathbf{a} = (-0.2, 1.0)$, is the same in all figures. Negative and positive contour lines are dashed and solid respectively.

5.2 Fold modelling

Modelling of geological folding is an important component in the Compound Earth Simulator. For this purpose Hjelle and Petersen [7] developed a mathematical framework that takes an horizon as initial condition and simulates a folded structure in a given domain. This framework can replicate all fold classes as defined by Ramsay [12] by changing parameter values of F and ψ in the equation

$$F\|\nabla T\| + \psi(\mathbf{a}\cdot\nabla T) = 1. \tag{5}$$

Here, $\mathbf{a}$ defines the axial direction of the fold. Interestingly, the characteristic curves to (5) coincide with the dip-isogons often used for classifications of folded structures. A minor extension of the algorithm is needed, since the folded structures above and below the initial horizon have a positive and negative axial direction respectively. This feature is modelled by propagating a sign with the front. The Fast Marching method does not create the correct solution for any but the parallel folding class, where $\psi = 0$. Figure 3 shows three folds from the same initial horizon (shown in red) where the dashed lines are negative isocurves, and solid lines positive. All folds in figure 3 have the axial direction $\mathbf{a} = (-0.2, 1)$, and are simulated on a fine grid of 401×401 nodes. For the class 1A fold in figure 3(a) we have $F = 1, \psi = -0.5$, for the class 1B (parallel) fold in figure 3(b) we have $F = 1, \psi = 0$, and for the class 1C fold of figure 3(c) we have $F = 1, \psi = 1$. The remaining fold classes 2 and 3 are modelled with $F = 0, \psi > 0$, and $F < 0, \psi > 0$, respectively.

5.3 Conclusions

We presented a new Semi-Ordered Fast Iterative algorithm for monotone front propagation. The SOFI method is fast and simple to implement, and works for general anisotropic front propagations. Unlike other iterative algorithms, the performance does not depend strongly on the variations of the speed, but somewhat on the degree of anisotropy. Instead, the performance of SOFI is more similar to the class of Ordered Upwind Methods, but no prior information on the degree of anisotropy or simplification of the algorithm are needed. For the SOFI method, numerical experiments indicate a computational scaling of degree $O(N)$, where N is the total number of unknown nodes. Regarding accuracy, the SOFI method has an identical solution to the Fast Marching method for isotropic equations, and as the Fast Sweeping method for anisotropic problems, when the methods has identical stencil formulations.

The SOFI method can use any consistent local wave approximation (stencil), and may therefore keep the constant velocity assumption to a small area. On isotropic examples the proposed method

was shown to be very fast compared to the Fast Marching method. This relation hods for both the more accurate diagonal stencil form, and the edge stencil. Huygens' principle implies that source points on the front that are not close to each other are independent of each other, thus implying parallel implementation possibilities of the SOFI method. However, there is a communication problem when different source points on different processors try to update the same node, most likely requiring a domain decomposition approach. Continuation of this work will compare the performance of the algorithm in three dimensions to the performance of other popular approaches, and further test efficiency of the method on anisotropic problems. It would also be interesting to test the algorithm on non-rectangular grids, where the method itself is directly applicable. Within seismic processing the computational time is often very long. Therefore the SOFI method is a promising alternative for simulating seismic traveltimes fast.

Acknowledgement

The author is very thankful to the anonymous reviewers who help to improve the paper. The presented work was funded by Statoil through the Akademia program, and the Research Council of Norway under grant 202101/I40. The work has been conducted at Kalkulo AS, a subsidiary of Simula Research Laboratory.

Bibliography

1. Alton, K. (2010). *Dijkstra-like Ordered Upwind Methods for Solving Static Hamilton-Jacobi Equations*. Ph. D. thesis, The University Of British Columbia.

2. Bak, S., J. McLaughlin, and D. Renzi (2010, September). Some improvements for the fast sweeping method. *SIAM Journal on Scientific Computing 32*(5), 2853–2874.

3. Berre, I., K. H. Karlsen, K.-A. Lie, and J. R. Natvig (2005, November). Fast computation of arrival times in heterogeneous media. *Computational Geosciences 9*(4), 179–201.

4. Cristiani, E. (2009). A fast marching method for Hamilton-Jacobi equations modeling monotone front propagations. *Journal of Scientific Computing 39*(2), 189–205.

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

6. Glover, F., D. Klingman, and N. Phillips (1985, January). A new polynomially bounded shortest path algorithm. *Operations Research 33*(1), 65–73.

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

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

9. Kim, S. (2001). An $O(N)$ level set method for eikonal equations. *SIAM Journal on Scientific Computing 22*(6), 2178–2193.

10. Qian, J., Y.-T. Zhang, and H.-K. Zhao (2007). A fast sweeping method for static convex Hamilton-Jacobi equations. *Journal of Scientific Computing 31*(1-2), 237–271.

11. Qin, F., Y. Luo, K. B. Olsen, W. Cai, and G. T. Schuster (1992, March). Finite-difference solution of the eikonal equation along expanding wavefronts. *Geophysics 57*(3), 478–487.

12. Ramsay, J. G. (1967). *Folding and fracturing of rocks*. McGraw-Hill, New York and London.

13. Rawlinson, N. and M. Sambridge (2004). Multiple reflection and transmission phases in complex layered media using a multistage fast marching method. *Geophysics 69*(5), 1338–1350.

14. Sethian, J. A. (1999). *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science*. Cambridge University Press.

15. Sethian, J. A. and A. Vladimirsky (2003). Ordered upwind methods for static Hamilton-Jacobi equation: Theory and algorithms. *SIAM Journal of Numerical Analysis 41*(1), 325 – 363.

16. Vidale, J. E. (1988). Finite-difference calculation of travel times. *Bulletin of the Seismological Society of America 78*(6), 2062–2076.

Paper III:
A new parallel 3D front propagation algorithm for fast simulation of geological folds

A new parallel 3D front propagation algorithm for fast simulation of geological folds

Tor Gillberg [1,2], **Mohammed Sourouri** [1,3], **Xing Cai** [1,3]

[1] Computational Geosciences, Simula Research Laboratory,
P. O. Box 134, N-1325 Lysaker, Norway

[2] Kalkulo AS,
P. O. Box 134, N-1325 Lysaker, Norway

[3] Department of Informatics, University of Oslo,
P. O. Box 1080 Blindern, N-0316 Oslo, Norway

Procedia Computer Science, 9, pages 947-955. Proceedings of the International Conference on Computational Science, ICCS 2012.

Abstract:

We present a novel method for 3D anisotropic front propagation and apply it to the simulation of geological folding. The new iterative algorithm has a simple structure and abundant parallelism, and is easily adapted to multithreaded architectures using OpenMP. Moreover, we have used the automated C-to-CUDA source code translator, Mint, to achieve greatly enhanced computing speed on GPUs. Both OpenMP and CUDA implementations have been tested and benchmarked on several examples of 3D geological folding.

1 Introduction

The arrival time of a propagating front is often described by non-linear static Hamilton-Jacobi equations. Advanced numerical algorithms are needed to efficiently compute solutions to those equations. It is therefore a challenge to implement fast solvers, especially for large 3D simulations. Solution algorithms are often divided into two groups, Front Tracking methods and Sweeping methods. Front Tracking methods [20, 24, 28] update node values in a strictly increasing order, and thus mimic a front expanding from the initial object Γ_0. These algorithms are sequential by construction, since the front passes only one node at a time. Front tracking methods for anisotropic propagation are known as Ordered Upwind Methods. These methods are complicated, and some must be simplified for implementation. Moreover, they often assume prior knowledge of the degree of anisotropy in the problem, see for instance [1, 6, 25]. Sweeping methods [19] compute the solution from a distance perspective by iterating over directions. This makes them faster than Front Tracking methods on simple problems [8]. In the cases with complex geometries or velocities that force the characteristics to be curved, the Sweeping methods are slow because many iterations are needed before convergence [5]. Since the iteration order is predetermined, Sweeping methods [31] are readily parallelized. However, the parallelism of the traditional Sweeping algorithms is limited, and the parallel speedup is therefore modest [14]. By an alternative formulation of the stencil and iteration order, abundant parallelism can be obtained as shown with the Parallel Marching Method (PMM) [29]. Early iterative algorithms trace the front in specific patterns such as expanding boxes [27], or by updating all nodes until convergence [23]. These can be made parallel as described in [17]. There are also a few algorithms that use concepts from both Front Tracking and Sweeping methods [2, 5, 7, 12].

To our knowledge, only two methods for front propagation have been successfully implemented on graphics processing units (GPUs), namely the Fast Iterative Method [12] and the PMM [29]. Of these two methods only the Fast Iterative Method is applicable in 3D, since the PMM was created for computing geodesic distances on surfaces. In this article, we present a new 3D algorithm with abundant parallelism, making the algorithm suitable for both multicore CPU and GPU architectures. Since we use the idea of an alternative stencil formulation from the PMM, we refer to our new algorithm as the 3D PMM. The 3D PMM has a highly parallel structure as nodes on an entire surface (planar cut of the 3D volume) can be updated in parallel. Moreover, thanks to the automated C-to-CUDA translator Mint [26], we can maintain an annotated serial C code for our 3D PMM method, without having to do low-level CUDA programming by hand. In comparison, the more general Fast Iterative method has a more intricate algorithmic structure, making CUDA programming much more involved.

1.1 Simulation of folds and other applications

Over the past several years, Statoil and Kalkulo have developed a novel paradigm for highly interactive modelling of complicated geological scenarios and processes. This methodology describes present-day geology as the realization of a series of geological events and processes along a geological timeline [4, 16]. Many processes rely on relevant surfaces and their corresponding metric property fields or maps like distances, gradients etc. [9]. These distance maps are described by the viscosity solution to the static Hamilton-Jacobi equation:

$$F\|\nabla T(\mathbf{x})\| + \psi(\mathbf{a}\cdot\nabla T(\mathbf{x})) = 1, \tag{1}$$

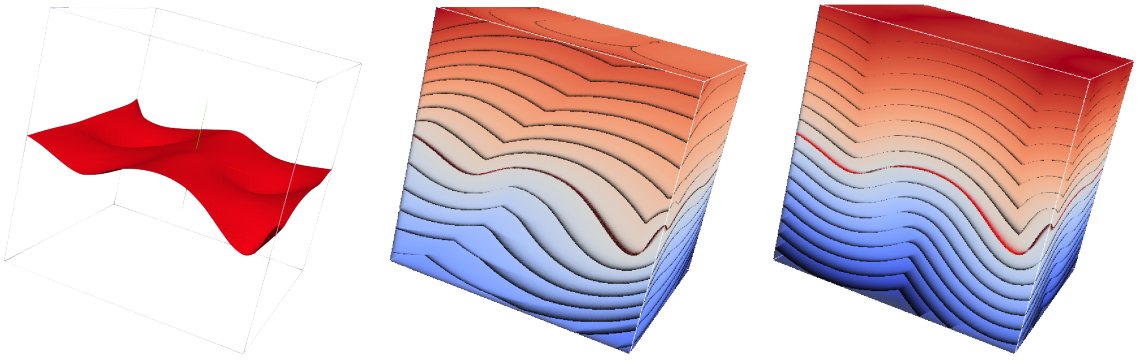
$$\text{given } T = t_0 \text{ on } \Gamma_0. \tag{2}$$

Here, Γ_0 is an initial horizon, and $\mathbf{a}$ marks the axial direction of the fold. Other folded layers are implicitly given as iso-surfaces of T, that is, the position of a front propagating from Γ_0 at different times. This equation can replicate all traditional folding classes, as defined by [21], by altering the

size and sign of F, ψ and direction of $\mathbf{a}$. For details and derivation of this system we refer to [9]. The same equation also describes the first arrival of a wave in a media in motion [13]. Figure 1(a) shows an example of an initial surface Γ_0. From this surface Figures 1(b) and (c) show two simulations of folds, created with different parameter choices. When $\psi \neq 0$, the front propagation is of the anisotropic type. In the special case where $\psi = 0$, (1) reduces to the isotropic eikonal equation, which is solved in many applications [24] and is isotropic in the sense that the velocity is independent of direction. When $F = 1$, the viscosity solution to the eikonal equation is the minimal Euclidean distance from Γ_0. The concept of characteristic curves or ray-paths is important in front propagations [9, 22]. These are curves along which a particle on the front is transported, and can also be interpreted as curves defining the shortest distance (fastest path) to Γ_0.

When modelling folds, the dip isogons, and thus the characteristic curves [9], are in general linear. For isotropic problems with linear characteristics Sweeping algorithms converge quickly [5], which motivates us to investigate related algorithms for the simulation of geologically folded structures. A powerful laptop is sufficient for an interactive geological modelling application in 2D, but the computational requirement is vastly higher in 3D. Other geological applications where the simulation of a propagating front is central include reservoir simulations [3] and simulation of seismic traveltimes [18, 22]. In seismic applications, front propagation solvers are used to simulate entire first arrival traveltime fields. By repeating the simulation from reflecting surfaces, multiple reflected traveltimes can be computed with front propagation solvers [10, 22]. Front propagation is also heavily used in medical imaging [11, 14, 15]. In these applications, the computational speed remains often a challenge. A faster solver would allow for more interactive applications, potentially leading to faster medical diagnoses and faster seismic processing. An interactive geological modelling application allows users to test many geological scenarios faster, leading to a better understanding of the inner earth. One method to achieve faster solvers is by making use of the powerful computational resources available in GPUs.

In this paper, we present a novel 3D front propagation algorithm, as well as a numerical stencil for solving (1). We also show how to parallelize the algorithm for both multicore CPU and GPU architectures. The parallelized codes are tested on several examples of geological folding, for which the parallelized codes scale well.



(a) Initial surface Γ_0 (b) Anisotropic propagation (c) Isotropic propagation

Figure 1: (a) An initially given surface Γ_0. (b) A folded 3D volume with $F = 1, \psi\mathbf{a} = \frac{1}{2}(-1,1,1)$, simulated from Γ_0. (c) A folded 3D volume from the same initial surface, but this time with $F = 1, \psi = 0$, resulting in isotropic front propagation and an Euclidean distance field.

2 The 3D Parallel Marching Method

Consider a 3D box grid with nodal values $T_{i,j,k}$ where $(1,1,1) \leq (i,j,k) \leq (n_x,n_y,n_z)$, and with a spacing of (dx,dy,dz). In this paper we assume that values at the nodes closest to Γ_0 are given, and all the other nodes are initially set to an infinite value. (Efficient methods for initializing such values are outside the scope of this paper.) In every iteration, a smaller T value is a better approximation, since we solve for the minimal distance (the first time of arrival). It is of great importance that a new approximation is not too small, since such values are never corrected. A methodology for assuring such a discretization of (1) is presented in A. The PMM iterates through the grid in axial directions, and computes new distance values based on nodal values along the iteration direction. In the x-direction, the 3D volume is first iterated in the increasing order of the i index, and then in the decreasing order of the i index. The same sub-sweeps are also repeated in the y- and z-directions. We refer to such a full iteration as a *sweep*, which consists of 6 *sub-sweeps* of the 3D domain. Pseudocode for the sub-sweeps for the x-direction is given below.

Algorithm 2.1: iSUBSWEEP()

comment: STENCIL() returns a new approximation (see Appendix A)

for $i \leftarrow 2$ **to** n_x

do { **for each** $j \leftarrow 1$ **to** n_y **and** $k \leftarrow 1$ **to** n_z in parallel

do { $t_{new} \leftarrow \text{STENCIL}(\mathtt{T}_{i-1,j\pm a,k\pm b}, a \in \{0,1\}, b \in \{0,1\})$

$\mathtt{T}_{i,j,k} \leftarrow \min\{t_{new}, \mathtt{T}_{i,j,k}\}$

for $i \leftarrow n_x - 1$ **downto** 1

do { **for each** $j \leftarrow 1$ **to** n_y **and** $k \leftarrow 1$ **to** n_z in parallel

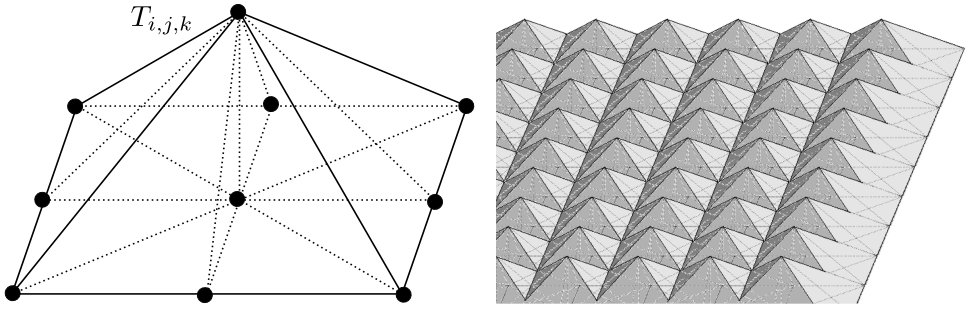
do { $t_{new} \leftarrow \text{STENCIL}(\mathtt{T}_{i+1,j\pm a,k\pm b}, a \in \{0,1\}, b \in \{0,1\})$

$\mathtt{T}_{i,j,k} \leftarrow \min\{t_{new}, \mathtt{T}_{i,j,k}\}$

We remark that $T_{i,j,k}$ is computed using nine nodes in the previously updated plane. The form of the update stencils are illustrated in Figure 2(a), where the sub-sweep is in the direction of the pyramid top. Every approximation's update step includes a significant amount of computations, as shown in A.

Since there are no internal dependencies between nodes on the same update plane, all nodes in the plane can be computed simultaneously. Figure 2(b) shows a plane of stencil shapes that can be solved in parallel. In the pseudocode this corresponds to computing the two inner loops in each sub-sweep entirely in parallel. Because of the simplicity of this parallelism, the algorithm is easily parallelized using OpenMP. For the OpenMP parallelization, the parallel region which encapsulates all the sweeps, is declared using **#pragma omp parallel**. Inside each of the six sub-sweeps, the two innermost nested loops are parallelized by adding **#pragma omp for**.

As shown in the next section, such a straightforward OpenMP parallelization achieves good speedup on multicore CPUs. Still, the OpenMP implementation is not sufficient for the application to be interactive for large grid sizes. This can be remedied by porting the algorithm to a GPU. To avoid manual GPU programming, we have made use of the automated C-to-CUDA source code translator Mint, freely available at https://sites.google.com/site/mintmodel/. Mint takes as input annotated serial C code and generates (optimized) CUDA code. The needed Mint pragmas are very similar to the OpenMP pragmas, except for two additional pragmas: **#pragma mint copy(T,toDevice,nx,ny,nz)** and **#pragma mint copy(T,fromDevice,nx,ny,nz)**, for transferring data between the host CPU and the device GPU. In Listings 1 and 2, we show two CUDA code segments that are automatically generated



(a) One stencil shape (b) Independent updates

Figure 2: (a) Nodes used when computing the generalized distance in the upward direction. (b) Illustration of the fine-grained parallel feature of the algorithm. All nodes in one plane can be computed simultaneously since they have no internal dependencies, making the algorithm suitable for parallel architectures.

by Mint.

Listing 1: The main computational body of the `Sweep` function after automated Mint translation from C to CUDA; The number of nodes in each spatial direction is 400.

```
for (int SweepNbr = 0; SweepNbr < nbrSweeps; SweepNbr++) {

  // x-direction: sweep from bottom to top
  for (i = 1; i < 400; ++i) {
    int num2blockDim_6_1527 = (400) % 16 == 0?(400) / 16 : (400) / 16 + 1;
    int num1blockDim_6_1527 = (400) % 16 == 0?(400) / 16 : (400) / 16 + 1;
    dim3 blockDim_6_1527(16,16,1);
    dim3 gridDim_6_1527(num1blockDim_6_1527,num2blockDim_6_1527);

    mint_6_1527<<<gridDim_6_1527,blockDim_6_1527>>>(DXYP,DXZP,DYZP,dev_1_T,i,tnew,
        st,xt,yt,txy,xnt,ynt,txm,tym,txnyn,F,ax,ay,az,dzz,dxx,dyy);
    cudaThreadSynchronize();
  }

  // x-direction: sweep from top to bottom
  // ...

  // y-direction: sweep from bottom to top
  // ...
  // y-direction: sweep from top to bottom
  // ...

  // z-direction: sweep from bottom to top
  // ...
  // z-direction: sweep from top to bottom
  // ...
}
```

Listing 2: The CUDA kernel function `mint_6_1527` that is automatically generated by Mint; for the purpose of sweeping one *yz*-plane.

```
__global__ void mint_6_1527(double DXYP,double DXZP,double DYZP,cudaPitchedPtr
    dev_1_T,int i,double tnew,double st,double xt,double yt,double txy,double xnt,
    double ynt,double txm,double tym,double txnyn,double F,double ax,double ay,
    double az,double dzz,double dxx,double dyy)
{
  double *T = (double *)dev_1_T.ptr;
  int _width = dev_1_T.pitch / sizeof(double );
  int _slice = dev_1_T.ysize * _width;
  int _p_j;
  int _p_k;
{
    int _upperb_y = 400;
    int _upperb_x = 400;
    int _idx = threadIdx.x + 1;
    int _gidx = _idx + blockDim.x * blockIdx.x;
    int _idy = threadIdx.y + 1;
    int _gidy = _idy + blockDim.y * 1 * blockIdx.y;
{
      if (_gidy >= 1 && _gidy <= 400) {
            if (_gidx >= 1 && _gidx <= 400) {
              // the same computations as in the original C code
            }
      }
}
}
}
```

3 Results

In this section we present numerical results from an example of simulating a folded volume. From the same initially given surface as in Figure 1, we ran 8 sweeps on the three uniform grids with a total of $160^3, 320^3$ and 400^3 nodes. After 8 sweeps the solution has converged sufficiently. In these computations $\psi a = (-0.34, 0.4, 0.7)$, $F = 1.1$, and the domain has length 10 in x, y and z directions. We have measured the computational time for the OpenMP code using one node on the NERSC Cray XE6 "Hopper" supercomputer. Each node is equipped with two twelve-core AMD 'Magny-Cours' 2.1 GHz processors. The Mint-translated CUDA code for the same problem was executed using a Nvidia GeForce GTX 590 card. Table 1 shows elapsed times for the three grids on $1, 2, 4, 8, 16$ and 24 CPU cores, as well as for the GPU. The time to transfer data to and from the GPU are included in the reported times. Both the CPU and GPU executables were compiled with the `-O3` flag, using `nvcc 4.0, V0.2.1221` and `gcc v4.3.4` respectively.

In A we present conditions that reduce the number of unnecessary computations in the update step. If all conditions are used, the number of branches increases. The update scheme already has many branches, as is indicated of profiling of the code. The profiling also indicate that the registers are under high pressure. Therefore, we have tried to formulate the update step to reduce unnecessary branching and register use. Several experiments was run with different update conditions, showing that the computational time is reduced the most when all conditions in A are used. This result holds for both the multicore and GPU versions of our code. Both the CPU and GPU codes was tested with both single and double precision. The difference between the single and double precision solutions is very small. Therefore, when modelling folds the gain in accuracy might not be worth the associated cost in computational time. Further investigation is needed before making any conclusion in this matter. In the geological modelling software, the derivative of the computed solution is used in post processes. This puts extra demand for high accuracy of the computed distance field. Thus, future research will be focused on extending the discretization to higher order schemes.

Table 1: Computational times for three grids with a total of N nodes using single (top table) and double precision. t_i is the CPU time for i cores. The speedup factors S_1 and S_{24} are calculated using the running time from 1 core and 24 cores (the highest number of CPU cores available). The speedup factor increases as N increases, possibly due increased computational complexity that amortises the data transfer cost from the CPU to the GPU. The data in Tabular (a) are single precision results while data in Tabular (b) are double precision results.

N	t_1	t_2	t_4	t_8	t_{16}	t_{24}	t_{GPU}	S_1	S_{24}
160^3	194.17	100.66	52.62	27.20	14.07	10.22	2.43	79.9x	4.2x
320^3	1795.86	822.48	423.87	219.39	112.57	81.40	14.84	121.0x	5.4x
400^3	3543.14	1628.12	853.50	430.59	223.55	177.86	25.28	140.1x	7.0x

(a) Computational times (t) and GPU speedup (S) for single precision

N	t_1	t_2	t_4	t_8	t_{16}	t_{24}	t_{GPU}	S_1	S_{24}
160^3	217.54	112.85	58.93	30.44	15.73	11.40	2.43	89.5x	2.4x
320^3	2016.50	911.12	472.84	245.08	125.08	90.07	31.09	64.8x	2.9x
400^3	3886.31	1799.22	928.89	481.99	246.80	240.05	57.42	67.6x	4.1x

(b) Computational times (t) and GPU speedup (S) for double precision

The code scales well on the multicore CPU, with near-linear speedup (1.9x) measured when conducting a strong scaling study up to 16 cores. Beyond 16 cores, the speedup drops to 1.3x. This drop in speedup is possibly due to the underlying NUMA (Non Uniform Memory Architecture) architecture on Hopper. With a more careful distribution of threads and data, we might be able to reduce the challenges NUMA imposes on the performance.

For the largest grid of 400^3 nodes, the GPU needs **25.28** seconds to perform 8 sweeps using single precision. As comparison, 24 CPU cores need **177.86** seconds to perform the 8 sweeps. When double precision is used on the GPU, the time usage is **57.42** seconds, more than **4** times faster than using 24 CPU cores (**240.05** seconds). It can also be seen in Table 1 that the speed advantage of GPU computations increases with the grid size.

At the moment of writing, a GeForce GTX 590 GPU costs around $500 USD, while one AMD 'Magny-Cours' 2.1 GHz costs more than $1000 USD. Depending on the grid size and precision, using a GPU will deliver 2-7x the performance, while costing[6] four times less than a 24-core CPU node. This makes the results even more impressive.

4 Discussions

Mint has been shown to deliver good performance on 3D finite difference codes [26]. Our algorithm is a 3D finite difference solver, but a non-traditional one. The grid is iterated in specific orders and a non-traditional stencil is used. Mint has delivered a surprisingly good GPU performance for our 3D PMM. A detailed comparison of the Mint-translated code with a hand-coded CUDA version would be an interesting investigation. We have experimented with the formulation of the update step, to search for an efficient formulation. Those optimization investigations indicate that the high number of branches introduced from conditioning the update computations, reduces the computational speed. Nevertheless, profiling has indicated some further optimization possibilities, for both CPU and GPU

[6] Other system parts such as memory, motherboard etc., are not taken into account.

implementations. For instance, the current register use is very extensive. A better use of the registers might improve both implementations.

An interesting algorithmic extension is to try ideas from [31], in which approaches for parallelization of the otherwise sequential Fast Sweeping Method are presented. One of the suggested ideas there is to sweep the domain in different directions at the same time on copies of the data structure, and then synchronize the results. Sub-sweeps in different directions can for instance be computed on several GPUs simultaneously. The approach of performing full sweeps of subdomains from the same paper may increase the convergence rate of the algorithm. Weber et al. [29] present a variant of the 2D PMM method to make it run faster on a GPU. Similar extensions in 3D are possible, and might assure good reuse of transferred data.

Although an $O(N)$ method [29], the PMM method often needs more Sweeps than the Fast Sweeping method to converge. Therefore, the algorithm is not suitable for applications with strongly curved characteristics, such as seismic data processing. For such applications the updates need to be ordered somehow. The Fast Iterative Method is one approach to this, but the ordering is too weak for complicated problems [12]. A related method exists for sequential 2D code on isotropic examples [5], in which subdomains are swept in a specified order. This idea can be extended to 3D and parallelized for GPUs by sweeping a list of subdomains simultaneously, using one streaming multiprocessor each, on which the streaming vector processors make use of the parallelism of the 3D PMM method. Furthermore, the subdomains can be ordered using an approach similar to that of [7] to ensure a stronger ordering, and convergence also on anisotropic problems. Such an algorithm will be efficient also on problems with bending characteristics.

5 Conclusion

Simulating a propagating front is a computationally challenging problem, especially in 3D applications. Simulations are needed in several applications, where the solution is needed within a few seconds for the software to be used in an interactive manner. In 3D, sequential algorithms are only applicable on small grid sizes. We have presented a simple 3D Parallel Marching Method and applied it to the simulation of geological folding. The algorithm can be easily implemented on parallel architectures. Numerical experiments using OpenMP show near-linear scaling on multicore CPUs. Using the automated C-to-CUDA code translator Mint, we obtained a CUDA implementation without manual GPU programming. The GPU implementation runs approximately 2.4-7 times faster than the fastest multi-threaded version on 24 CPU cores, giving hope to compute large 3D grids interactively in the future.

6 Acknowledgments

The presented work was funded by Statoil through its Academia Program, and the Research Council of Norway under grant 202101/I40. The work has been conducted at Kalkulo AS, a subsidiary of Simula Research Laboratory.

A Conditional upwind approximations

As in most front propagation methods, it is of great importance that approximations are computed from upwind values. Upwind values are values that are passed by the front. Monotonic convergence is a fundamental property for convergence toward the viscosity solution for most algorithms [12, 23, 24]. A too small approximation will not be increased, since that would contradict the monotonicity assumption.

Therefore, one must assure that the computed value uses solution values that are upwind from the updated node. We assert this by computing the characteristic curve of the approximation, and make sure that it is embedded in the convex hull of the nodes used in the computations. If it is not, the new approximation is rejected. A similar approach for isotropic problems are presented in [30], where the entrance point is used to find the rays in a seismic processing setting. From [9] we have the characteristics $x(s)$ to (1)

$$\frac{\partial x}{\partial s} = F\nabla T + \psi \mathbf{a}|\nabla T|. \tag{3}$$

Consider the stencil shape as given in Figure 2(a), where a new solution is sought for the pyramid-top node value $T_{i,j,k}$, and the nine nodes in the lower plane all have the third coordinate as $k-1$. The nodes on the lower plane are divided in eight groups of three nodes that form trirectangular tetrahedras with $T_{i,j,k}$ as apex. Similarly, 16 groups of two nodes forming right-angled triangles are created (dotted lines on the lower plane) as well as nine groups of one node each. From each of these groups solution estimates are created. If $T_{i,j,k}$ is bigger than the smallest acceptable of these approximations, we update the value at (i,j,k) with the new estimate. With estimates of the gradient of T and equation (3), the entrance point in the lower plane is given as

$$x_e = -dz\frac{F\frac{\partial T}{\partial x} + \psi \mathbf{a}_x|\nabla T|}{F\frac{\partial T}{\partial z} + \psi \mathbf{a}_z|\nabla T|}, \quad \text{and} \quad y_e = -dz\frac{F\frac{\partial T}{\partial y} + \psi \mathbf{a}_y|\nabla T|}{F\frac{\partial T}{\partial z} + \psi \mathbf{a}_z|\nabla T|}. \tag{4}$$

Assume the nodes $T_{i,j,k-1}, T_{i+1,j,k-1}$ and $T_{i+1,j+1,k-1}$ are three nodes in a trirectangular shape. With these nodes we estimate the partial derivatives in x and y direction with

$$\frac{\partial T}{\partial x} = \frac{T_{i+1,j,k-1} - T_{i,j,k-1}}{dx}, \quad \text{and} \quad \frac{\partial T}{\partial y} = \frac{T_{i+1,j+1,k-1} - T_{i+1,j,k-1}}{dy}. \tag{5}$$

Using these two estimates, we directly discretize (1) and get $\frac{\partial T}{\partial z}$ as the solution of a second degree polynomial. From (4) we get the entrance coordinates x_e, y_e, and the solution estimate $T^{new}_{i,j,k} = T_{i,j,k-1} + dz\frac{\partial T}{\partial z}$ is accepted if

$$0 < \min x_e, y_e, \quad y_e dx < x_e dy, \quad \text{and} \quad x_e < dxx. \tag{6}$$

That is to assure the entrance point is within the convex hull of the nodes used in the stencil. The remaining stencils using three values, are identical up to a rotation and reflection.

For the two node group using $T_{i,j,k-1}$ and $T_{i+1,j,k-1}$ we estimate $\frac{\partial T}{\partial x}$ with $\frac{T_{i+1,j,k-1}-T_{i,j,k-1}}{dx}$. That the characteristic curve to $T_{i,j,k}$ cut the line segment between $(i,j,k-1)$ and $(i+1,j,k)$ is to say that $y_e = 0$ of (4), that is

$$\frac{\partial T}{\partial y} = \frac{-\psi a_y|\nabla T|}{F}. \tag{7}$$

Together with (1) we get $\frac{\partial T}{\partial z}$ as the solution to a second degree polynomial. If the entrance point x_e from (4) satisfies $0 < x_e < dx$, we accept the new approximation. The remaining 15 stencils using two values are identical up to a rotation and reflection.

When only the value at $T_{i,j,k-1}$ is used, the characteristic curve must go from $(i,j,k-1)$ through (i,j,k). That is $x_e = 0, y_e = 0$ in (4), resulting in

$$F\frac{\partial T}{\partial x} + \psi a_x|\nabla T| = 0, \text{ and } F\frac{\partial T}{\partial y} + \psi a_y|\nabla T| = 0. \tag{8}$$

The traveltime solution for this system is easiest found using the group velocity v_G, that is the velocity in the direction of motion [19, 28]. In our case we have the the group velocity in the z-direction as $v_G = F\frac{\frac{\partial T}{\partial z}}{|\nabla T|} + a_z$, and the arrival time to $T_{i,j,k} = T_{i,j,k-1} + \frac{dz}{v_G}$. For a general point with value $T_{l,m,n}$ at the distance $\mathbf{x} = (x,y,z)$ from index (i,j,k) the corresponding solution is

$$T_{i,j,k} = T_{l,m,n} + \frac{\mathbf{x}\cdot\mathbf{x}}{\mathbf{a}\cdot\mathbf{x} + F\sqrt{(1-\frac{|\mathbf{a}|^2}{F^2})|\mathbf{x}|^2 + \frac{(\mathbf{a}\cdot\mathbf{x})^2}{F^2}}}. \tag{9}$$

In 2D this result is the same as the analytical solution of a point source as shown in [13].

Reducing the number of redundant computations

In isotropic front propagations, only strictly smaller nodes should be used for creating solution estimates with upwind stencils [24]. For a general anisotropic problem the same principle does not hold. However, at least one of the used nodes must be smaller than the old estimate for there to be a possibility of an acceptable solution estimate [7]. In the above stencil formulation this correspond to $T^{old}_{i,j,k} > \min_{(a,b)\in\{(0,0),(1,0),(1,1)\}} T_{i+a,j+b,k-1}$ in the three node case, $T^{old}_{i,j,k} > \min_{a\in\{0,1\}} T_{i+a,j,k-1}$ for the two node case, and $T^{old}_{i,j,k} > T_{i,j,k-1}$ for the one node case.

Moreover, we can derive the following condition for the characteristics entrance point x_e and y_e to be grater than 0

$$n_x = \frac{\frac{\partial T}{\partial x}}{|\nabla T|} < \frac{a_x}{F}, \quad \text{and} \quad n_y = \frac{t_y}{|\nabla T|} < \frac{a_y}{F}. \tag{10}$$

If $a_x < 0$ then we must have $\frac{\partial T}{\partial x} < 0$, otherwise the solution will not be accepted, and hence we need not to create the estimate. The corresponding argument holds for the y_e entrance point.

Remark; signed distance

When simulating folds, one must distinguish between the inside and outside of the structure. In order for the fold to be consistent, the axial direction $\mathbf{a}$ is negative on the inside, and positive on the outside. The same holds for the solution T. Accordingly, the 3D PMM adjusts the initialisation step slightly. The initialised data is set to $-\infty$ on the inside, and $+\infty$ on the outside. The sign of $T_{i,j,k}$ during the update step is saved locally, and the update is performed with absolute values of the nodes. If a new solution is found, it is saved with the same sign as the previous approximation was.

Bibliography

1. Alton, K. (2010). *Dijkstra-like Ordered Upwind Methods for Solving Static Hamilton-Jacobi Equations*. Ph. D. thesis, The University Of British Columbia.

2. Bak, S., J. McLaughlin, and D. Renzi (2010, September). Some improvements for the fast sweeping method. *SIAM Journal on Scientific Computing 32*(5), 2853–2874.

3. Berre, I., K. H. Karlsen, K.-A. Lie, and J. R. Natvig (2005, November). Fast computation of arrival times in heterogeneous media. *Computational Geosciences 9*(4), 179–201.

4. Bruaset, A. M. (2010). *Simula Research Laboratory – by thinking constantly about it*, Chapter Turning Rocks into Knowledge, pp. 553–600. Springer.

5. Chacon, A. and A. Vladimirsky (2012). Fast two-scale methods for eikonal equations. *SIAM Journal on Scientific Computing 34*(2), A547–A578.

6. Cristiani, E. (2009). A fast marching method for Hamilton-Jacobi equations modeling monotone front propagations. *Journal of Scientific Computing 39*(2), 189–205.

7. Gillberg, T. (2011). A semi-ordered fast iterative method (SOFI) for monotone front propagation in simulations of geological folding. In *MODSIM2011, 19th International Congress on Modelling and Simulation*, pp. 641–647.

8. Gremaud, P. A. and C. M. Kuster (2006, December). Computational study of fast methods for the eikonal equation. *SIAM Journal on Scientific Computing 27*(6), 1803–1816.

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

10. Huang, J.-W. and G. Bellefleur (2012). Joint transmission and reflection traveltime tomography using the fast sweeping method and the adjoint-state technique. *Geophysical Journal International 188*(2), 570–582.

11. Jeong, W.-K., P. T. Fletcher, R. Tao, and R. Whitaker (2007, November – December). 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.

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

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

14. Li, S., K. Mueller, M. Jackowski, D. Dione, and L. Staib (2008). Physical-space refraction-corrected transmission ultrasound computed tomography made computationally practical. In *Medical Image Computing and Computer-Assisted Intervention - MICCAI 2008*, Volume 5242 of *Lecture Notes in Computer Science*, pp. 280–288. Springer Berlin / Heidelberg.

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

16. Petersen, S. A. and Ø. Hjelle (2008). Earth recursion, an important component in shared earth model builders. *EAGE 70th Conference & Exhibition, Extended Abstracts*.

17. Podvin, P. and I. Lecomte (1991). Finite difference computation of traveltimes in very contrasted velocity models: a massively parallel approach and its associated tools. *Geophysical Journal International 105*, 271–284.

18. Popovici, A. M. and J. A. Sethian (2002). 3-D imaging using higher order fast marching traveltimes. *Geophysics 67*(604, Issue 2), 604 – 609.

19. Qian, J., Y.-T. Zhang, and H.-K. Zhao (2007). A fast sweeping method for static convex Hamilton-Jacobi equations. *Journal of Scientific Computing 31*(1-2), 237–271.

20. Qin, F., Y. Luo, K. B. Olsen, W. Cai, and G. T. Schuster (1992, March). Finite-difference solution of the eikonal equation along expanding wavefronts. *Geophysics 57*(3), 478–487.

21. Ramsay, J. G. (1967). *Folding and fracturing of rocks*. McGraw-Hill, New York and London.

22. Rawlinson, N. and M. Sambridge (2004). Multiple reflection and transmission phases in complex layered media using a multistage fast marching method. *Geophysics 69*(5), 1338–1350.

23. Rouy, E. and A. Tourin (1992). A viscosity solutions approach to shape-from-shading. *SIAM J. Numer. Anal. 29*(3), 867–884.

24. Sethian, J. A. (1999). *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science*. Cambridge University Press.

25. Sethian, J. A. and A. Vladimirsky (2003). Ordered upwind methods for static Hamilton-Jacobi equation: Theory and algorithms. *SIAM Journal of Numerical Analysis 41*(1), 325 – 363.

26. Unat, D., X. Cai, and S. Baden (2011). Mint: Realizing CUDA performance in 3D stencil methods with annotated C. In *Proceedings of the 25th International Conference on Supercomputing (ICS'11)*, ICS '11, New York, NY, USA, pp. 214–224. ACM.

27. Vidale, J. E. (1988). Finite-difference calculation of travel times. *Bulletin of the Seismological Society of America 78*(6), 2062–2076.

28. Wang, Y., T. Nemeth, and R. T. Langan (2006). An expanding-wavefront method for solving the eikonal equations in general anisotropic media. *Geophysics 71*(5), T129–T135.

29. Weber, O., Y. S. Devir, A. M. Bronstein, M. M. Bronstein, and R. Kimmel (2008). Parallel algorithms for approximation of distance maps on parametric surfaces. *ACM Transactions on Graphics (TOG) 27*(4), 104:1–104:16.

30. Zhang, J., Y. Huang, L.-P. Song, and Q.-H. Liu (2011, March). Fast and accurate 3-D ray tracing using bilinear traveltime interpolation and the wave front group marching. *Geophysical Journal International 184*(3), 1327–1340.

31. Zhao, H.-K. (2007). Parallel implementations of the fast sweeping method. *Journal of Computational Mathematics 25*(4), 421 – 429.