ZERO

Accuracy in Scientific Visualization

by

Adriano Martins Lopes

Submitted in accordance with the requirements

for the degree of Doctor of Philosophy.

The University of Leeds

School of Computer Studies

March 1999

The candidate confirms that the work submitted is his own and the appropriate credit has been given where reference has been made to the work of others.

Summary

Quite often, accuracy is a negleted issue in scientific visualization. Indeed, in most of the visualizations there are two wrong assumptions: first, that the data visualized is accurate. Second, that the visualization process is exempt from errors. On these basis, the objectives of this thesis are three-fold:

First, to understand the implications of accuracy in scientific visualization. It is important to analyse the sources of errors during visualization, and to establish mechanisms that enable the characterization of the accuracy. This learning stage is crucial for a sucessful scientific investigation.

Second, to focus on visualization features that, besides enabling the visualization of the data, give users an idea of its accuracy. The challenging aspect in this case is the use of appropriate visual paradigms. In this respect, the awareness of how human beings create and process a mental image of the information visualized is important.

Thrid and most important, the development of more accurate versions of visualization techniques. By understanding the issue of accuracy concerning a particular technique, there is a high probability to reach to a proposal of new improvements.

There are three techniques under study in this thesis: contouring, isosurfacing and particle tracing. All these are widely used in scientific visualization. That is why they have been chosen. For all of them, the issue of showing accuracy to users is discussed. In addition, two new accurate versions of contouring and isosurfacing techniques have been presented. The new contouring method is for data defined over rectangular grids and assumes that the data vary linearly along the edges of the cell. The new isosurfacing method is an improvement of the Marching-Cubes method. Some aspects of this classic approach are clarified, and even corrected.

Sections of this work were presented in the Visualization and Mathematics 97 Conference, held in Berlin, Germany, in September 1997, and in the Eurographics UK 98 Conference, held in Leeds, United Kingdom, in April 1998.

Acknowledgments

I acknowledge with deep gratitude the guidance, interest, and friendship provided by my supervisor Dr Ken Brodlie. I am also grateful to his wife Trish and family.

I also would like to thank Dr José Carlos Teixeira and Dr Helen Wright for the valuable support and guidance provided.

I am grateful to all the colleagues from the Scientific Visualization Group: Jason Wood, Nuha El-Khalili, Rafiq Asim, Stuart Lovegrove, Yasmin Aziz and Ying Li, for their help and friendship. This is extended to Elyas Nurgat, Huda Al-Foudri, Rasha El-Khalili and Samantha Henderson.

I also acknowledge the help of Dr Martin Berzins, Dr Tim David, Dr Jeremy Walton and Dr Richard Morris throughout the research undertaken.

Also, many thanks to Dr Martins Vaz, Vitor Murtinho, Lucinda Ferreira, Cantador Marques, Joaquim Madeira, Francisco Leite, Jorge Guiomar and João Sebastião for their help and friendship.

I thank the many friends from the Department that made me feel as if I was at home. Special mention goes to my fellow football players.

I acknowledge the support of the Portuguese Programme PRAXIS XXI, under contract BD / 2756 / 94, and the Department of Mathematics of the University of Coimbra.

Finally, special gratitude goes to my parents and my family for their care and encouragement.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter

ONE

Introduction

Visualization is nowadays accepted as a crucial part of scientific and engineering computation. Although not a new idea, it has gained new meaning and importance over recent years. The more society becomes computer and multimedia oriented, the bigger the role scientific visualization has. There are plenty of examples that confirm this expanding trend. We can show medical information such as three-dimensional representation of parts of the human body based on magnetic resonance imaging, allowing us to analyse living organs non-invasively. We can analyse environmental data like tornadoes and ocean currents, with application for example in the study of global warming and the ozone layer. We can generate cartographic maps, and analyse geological data with application in the recovery of oil from deep water. We can plot statistical data such as prices of securities listed in a stock market exchange, in this case to eventually help investors to predict future movements, *etc.*

More than displaying data graphically, scientific visualization enables scientists to observe and understand the data that characterise their problems. Its success is particularly due to the ability to convey large amounts of data as images that subsequently, as compact pieces of information, are directly read by a powerful human sense: the vision. The creation of a mental image of the solution of a problem is therefore fast. Indeed, to

see means to actively capture prominent characteristics of objects.

This is also applied to situations where there is a need for scientists to transfer their knowledge to others in general. To cite an example, let us consider the analysis of distribution of public investments in a certain local county — hospitals, schools, transport networks, *etc.* Methods of operational research are used to obtain solutions for the problem, which means there are large amounts of data (relationships and constraints) involved. Most certainly the solutions are better understood by decision makers if conveyed pictorially. These people, not necessarily familiar with concepts of operational research, simply would not be able to analyse crude tables of data.

Yet, the images produced cannot always be trusted. Accuracy is indeed an aspect quite often ignored in visualization. A scientist inputs their data into a visualization system. Then he or she analyses the visualizations created. This is a typical scenario in scientific investigation. In this process, there are two aspects to note: first, the data that represents the problem is assumed to be accurate; secondly, the scientist assumes that the visualization produced is an accurate portrayal of the data. In reality both assumptions are false: the data is subject to error; the visualization process is subject to error. Therefore, if uncertainties are not taken into consideration, then the visualization can be misleading and so affect the credibility and validity of the scientific analysis.

The issue of accuracy in scientific visualization is the underlying subject of study in this research. The objectives we set out are three-fold:

First, we aim to understand, in general, the implications of accuracy in scientific visualization. We need to understand the requirements of visualizing error information associated with the data. It implies the need to understand the sources of error throughout the visualization process, how they interact with each other, and we need to establish methods that enable us to obtain statements about the accuracy of data, most certainly by means of quantitative metrics. We believe that this learning stage is crucial for the success of the research undertaken.

Second, we aim to develop techniques that show the accuracy of the visualizations produced. Both data and associated error data should be presented to the users, whether combined in the same picture or not. Ultimately scientists should visualize not only the

data but also a representation of the assumptions that have been made during the creation of that visualization. Here the challenging aspect is the use of appropriate visual paradigms. It is important to be aware of the way the visual information is processed by the human vision, and more generally the cognitive understanding.

Finally, and what is very important, we are interested in developing more accurate versions of certain visualization techniques. It is our belief that a full understanding of the issue of accuracy concerning a particular technique will eventually lead to the proposal of improvements. By doing so we will contribute to the development of scientific visualization as a discipline.

We should point out that we confine ourselves, in the visualization process, to the mapping of the input data into geometries, that subsequently are rendered as images. On that basis, our efforts are concentrated on three visualization techniques. They are:

1. Contouring, to draw contour lines in a two dimensional scalar data set.
2. Isosurfacing, to represent surfaces with points of equal scalar value, in a three dimensional data set.
3. Particle tracing, to show the trace of particles within a three dimensional vector data set.

This choice is related to the importance of these techniques in the context of scientific visualization. All are widely used. The first two deal with scalar data whereas the last one with vectorial data.

1.1 Thesis Organisation

Apart from this introductory Chapter, this thesis is organised in the following way:

Chapter Two — Scientific Visualization — offers an overview of scientific visualization as a discipline. The aim is not only to provide the basic foundations that further Chapters will rest on, but also give readers a general understanding of what scientific visualization is all about.

Chapter Three — Contouring — covers the technique of contouring scalar data in two dimensional space. First, the method is discussed, in particular with methods that work on rectangular grids, and such as the data is assumed to vary linearly between adjacent data points. From the analysis there comes out a new proposal, relying upon a basic method but more accurate. This new approach is compared with similar ones. The Chapter ends by addressing the problem of showing accuracy in contouring, with application to those methods under study. Part of this work on contouring was presented in the Eurographics UK 98 Conference, held in Leeds, United Kingdom, in April 1998.[66]

Chapter Four — Isosurfacing — discusses the technique of drawing surfaces of constant value over a three dimensional scalar data set. The major feature in this Chapter is the presentation of a new accurate Marching-Cubes method. A full description is presented. But to reach that stage, a deep analysis of Marching-Cubes is carried out first. As a result, some aspects of this traditional and widely used method were clarified, and indeed, corrected. The Chapter ends with some hints about showing the accuracy of the visualizations produced with this new method.

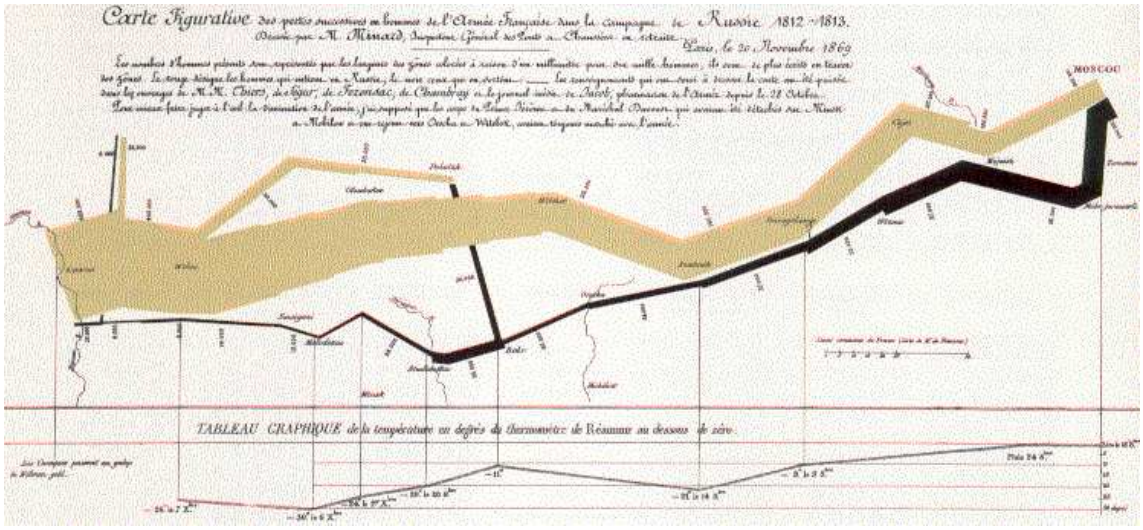
Chapter Five — Particle Tracing — aims to show the accuracy of particle tracing, a basic technique for vector data. After a general overview of the method, some strategies to quantify the error committed by the integration technique are presented. An architecture is established which is then followed by implementation. This work was presented in the Visualization and Mathematics 97 Conference, held in Berlin, Germany, in September 1997.[65]

Chapter Six — Conclusions and Future Work — summarises the work undertaken and identifies potential areas for further research.

Chapter

TWO

Scientific Visualization



The Russian campaign of Napoleon in 1812-1813, by Charles Minard. The army size is indicated by the brownish (advance) and black (retreat) bands. Reproduced from Tufte,[103] page 176.

From the early days of mankind the image has been used as a powerful tool of communication. It comes as no surprise since it targets the vision, the human sense of highest perception, with a large portion of brain devoted to it.

In this chapter we start with a historical view of visualization, finishing with current scientific visualization concepts, such as the dataflow model. Next, we lay down some aspects of an important task in visualization: the organization of data that is about to

be processed. Then we draw attention to the issue of accuracy in visualization, since we believe accurate visualization is crucial for scientists. Finally we present an overview of common visualization techniques, covering their range of applicability from scalar, to vector and tensorial data.

2.1 Historical View

Back in the pre-historic era, towards the final palaeolithic age (around 20,000 years ago), primitives painted and engraved figures of animals in rocks [a]. It is generally believed that they were associated with magical hunting rites as a means of impressing and strengthening authority of a few over the community.[75]

Many ancient civilizations have continued to use the image as a mechanism of expression. Egyptians, Byzantines, all used it to edificate the faithful and engrain religious lore. In the Renaissance period in Europe, there was a move from symbolic expression of theological truths towards a mundane and scientific approach. An example of this is the invention of perspective, supposedly by Brunelleschi in the beginning of the 15th century. It still remains a powerful technique of representation, giving the possibility to see different objects in a correct geometric context.

At the same time, fields such as astronomy, meteorology and cartography were demanding new methods of representation. Mostly, sailors needed accurate aids to navigate and discover the new worlds. A combination of cartographic and statistical skills joined together, although it has been suggested that a few thousand years before the Chinese already drew geographic maps on clay tablets [b].

There are very good examples of visual representations made during the period 15th-19th century (see Tufte,[103,105] and Collins[20]). For instance, the map of the cholera epidemic in London, and a representation of the army of Napoleon during the Russian campaign, both from the 19th century, are widely known examples of the usefulness and transparency of scientific visualization.

[a] Examples are found in *Altamira* (Spain) and *Lascaux* (France) and recently in *Foz-Côa* (Portugal).

[b] For example, some 900 years ago a fully scaled map of part of China (map of *Tracks of Yu the Great*) was engraved by Chinese carthographers.[20,103]

The map of the cholera epidemic was drawn by Dr. John Snow who plotted the location of deaths from cholera in central London in 1854. This scatter plot helped him to conclude that the cholera epicentre was near the Broad Street water pump. Once that pump of contaminated water was removed, the epidemic that had claimed more than 600 lives, ended.[103,105] It is a scientific visualization where data was indeed represented in an appropriate context to assess cause and effect.

The second example is depicted in the Figure at the beginning of this Chapter, on page 5. Therein, the French engineer Charles Minard portrays clearly the fate of the army of Napoleon during the Russian campaign in 1812-1813. He plotted the size of the army, its location, the direction of movement, and the temperature at various locations during the retreat. The width of the bands indicate the size of the army from place to place. The black band shows the retreat, linked to the temperature. Also shown are the movements of auxiliary troops to protect the rear and the flank of the advancing army. Napoleon had started the campaign with 442,000 men and returned to Poland with merely 10,000 men.[69,103]

Hence, scientific visualization is not a recent field. However, the computer revolution has broadened its scope of application. In the 1940s and 1950s, maps started to be produced by computer, as well as other data plots. More and more scientists were using computers to simulate and study their problems. The amount of data collected and/or generated was such that it demanded a greater effort to assimilate and understand it. Once more, vision is the sense *par excellence* to gain insight into data. Computer Graphics as a scientific field emerges. Libraries of graphics routines such as UNIRAS and NAG Graphics were used alongside numerical routines,[9] in this era of batch computing. This was extended in the late 1970s to menu-driven packages such as *gnuplot* where there was no longer a need for user programming, in accordance with the era of interactive computing, with terminals linked to a host. At this time graphical representation included just one or two variate data. The first steps towards graphics standardization were made with the Graphical Kernel System (GKS).[48] Then attention was gradually centred around encoding multiple parameters in the same picture.[122] Furthermore, people started to

become aware of techniques for a better visual decoding of information, such as those found in works of Gibson,[38] Tukey,[106] Bertin,[4] Tufte[103,104] and Cleveland.[18]

The move towards visualization was gaining momentum. In 1987, the ACM SIGGRAPH report *Visualization in Scientific Computing*, by McCormick *et al.* ,[71] marks a turning point in scientific visualization. It argues that individual disciplines such as computer graphics, image processing, computer vision, computer-aided design, signal processing and human computer interaction all contribute to the visualization process. After all, scientists have been doing visualization all the time. While computer graphics creates images, visualization should encompass exploring, transforming and viewing data, all to promote a better understanding of the data. The report also emphasises the need for both two and three dimensional spatial visualization. After that report, the research interest on the field has spiralled to new heights. Conferences such as IEEE Visualization in United States, and Eurographics Workshops on Visualization in Europe, become regular events yearly. A major breakthrough was the development of a new type of visualization system, the so-called Modular Visualization Environments (MVEs). Examples are AVS, Khoros, IBM Data Explorer and IRIS Explorer. It was the era of visual programming systems, with workstations and graphical user interfaces.

In the meantime in Computer Graphics, Programmer's Hierarchical Interactive Graphics System (PHIGS) emerged, while GKS was moving to three dimensions (GKS-3D). Also, the OpenGL library and Open Inventor toolkit became a de facto industry standard for graphical rendering on workstations.

The MVEs are mainly based on a model introduced by Upson *et al.* ,[108] and then enhanced by Haber and McNabb.[43] The visualization process is divided into three major sequential stages. They are:

Filtering.

In this first stage, the entered raw data is filtered in order to obtain the data of interest [c]. For instance, extracting a cross-section of data, or interpolating scattered data in order to obtain a regular grid.

[c] Sometimes the general concept of *modelling* is used instead.

Mapping.

In this second stage, the aim is to convert the data of interest into geometric representations. For example, computing the positions of a particle released in a flow field, or finding out the geometric surface of points of equal value.

Rendering.

Finally the geometries are converted into images.

In general, a visualization system such as IRIS Explorer provides a library of modules, matching one of those three stages above. Scientists then have to choose a suitable set of modules and link them as in a network (so-called *map of modules*). Data is fed into the network, then *flows* through it until it reaches the rendering modules. In addition, it is possible for users to construct their own modules thus increasing the functionality of the environment.

This data-flow model is still in use today, despite being a data-centered approach, ignoring the cyclic nature of scientific investigation.[2] New paradigms are emerging so that it is expected to see the visualization model being adjusted. While visualization used to be limited to one person at a time, it now tends to become more collaborative.[123,125] Scientists with diverse and complementary skills will team up to solve common problems. Events such as the massive explosion of World Wide Web (WWW) are affecting the way scientific visualization has been conceived in the dataflow model. Work by Wood and colleagues[123,124] suggests this direction. In one of their examples, air quality data is collected hourly throughout the United Kingdom and then visualized on demand by users. As a matter of a fact, current MVEs such as AVS and IRIS Explorer are now being adjusted to this new reality.

Another technological edge affecting visualization concepts is the revolutionary interaction paradigm of Virtual Reality (VR). There are a few examples of how to use VR to allow scientists to interact with the visualized data in 3D space. That is the case of the virtual wind tunnel for flow visualization.[12] The more VR technology improves the more likely it will be used in scientific visualization. Indeed, in years to come VR fulfills the

potential of getting rid of the burden of interaction and allow users to concentrate more on data. More and more we expect computers to adapt to users rather than the opposite.

We would like to stress as a final point that, regardless of how diverse users are or how impressive new technologies can be, the image is and will be always universal.

2.2 Data Taxonomy

The raw data describing a problem, whether simulated or acquired, is usually defined at a limited number of points within the domain. This collection of data values is usually referred as a data grid. If one wants to obtain values in-between the given points, then an interpolation method is required. This operation fits into the data preparation stage of the dataflow model.

We consider the taxonomy for data grids as depicted in Figure 2.1. First, a grid can be classed as structured or unstructured. In a structured grid, the constituents elements (cells) are topologically equivalent to a square (2D) or a cube (3D). The connectivity among nodes is completely defined by the nodes indices. But if there is no logical organization in the grid then it is called unstructured.

For a structured grid, if a square is deformed then we use the term quadrilateral whereas in case of a deformed cube the term is hexahedron. A structured grid can be rectilinear (also named rectangular) or curvilinear, a classification based upon the type of domain: orthographic and non-orthographic respectively. Moreover, a rectlinear grid is usually named as regular if the cells are evenly spaced and Cartesian (uniform) if that spacing is always constant regardless of the coordinate direction.

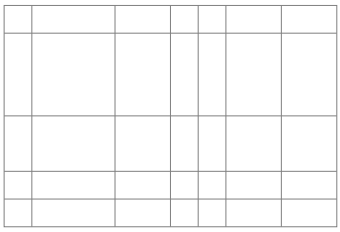
Finally, we would like to draw attention to the fact that different taxonomy may be found in the literature.

2.3 Accuracy of Visualizations

Most visualization techniques, no matter well outlined they be, take for granted that the underlying data is accurate. Unfortunately, as mentioned earlier, that is not always the

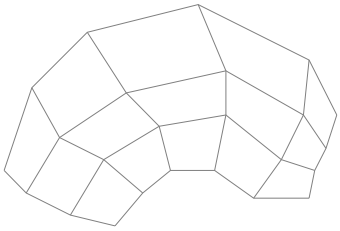
Rectilinear (also named rectangular).

Orthographic domain.
Cells non-evenly spaced.
Implicit coordinates.



Curvilinear.

Non-orthographic domain.
Cells non-evenly spaced.
Explicit coordinates.



Unstructured.

Explicit topology.
Adjacency of cells computed.
Explicit coordinates.
Usually in triangle sections.

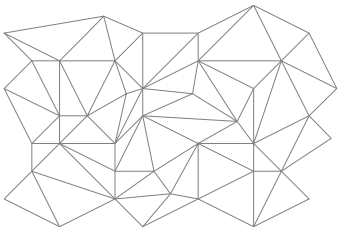


Figure 2.1: Taxonomy of data grids: rectilinear (or rectangular), curvilinear and unstructured. We should point out that a rectlinear grid is usually named as regular if the cells are evenly spaced, and Cartesian (or uniform) if that spacing is constant regardless of the coordinate direction.

Figure 2.2: Múndi-map in 1519 by Lopo Homem. The Earth was supposed to be completely discovered, so Europe and the Mediterranean Sea were largely magnified. Reproduced from Gil.[39]

case. Many examples can be shown where truth is a mirage. Let us just indicate two cases of erroneous visualizations made in the 16th century. The first one is a múndi-map as depicted in Figure 2.2. It is a synoptical vision of the Earth, a place *admired from top* where everything was already discovered.[39] The Mediterranean Sea is falsely perceived as bigger than it was, including the Black and Caspian Seas. It separates a large Europe from a small Africa.

The second one is more bizarre. Albrecht Dürer in 1515 produced an engraving that portrays a zoologically inaccurate rhinoceros.[101,105] It was to remain as a standard model for quite a long time (around 200 years), being repeatedly copied in subsequent works on travel and natural history. It was even incorporated into a monument [d].

Moving now to current days, the problem of accuracy in visualization is still important. Researchers from the Geographical Information Systems (GIS) field have devoted some attention to this issue, particularly in Cartography. In parallel to the examples above, most of their concerns were related to the accuracy of the data itself, not the visualization

[d] King Emmanuel I of Portugal sent an embassy to Rome to impress the Pope Leo X by offering him an elephant, a jaguar and a rhinoceros. Unfortunately the vessel carrying the rhinoceros, which was already a gift from King Muzafar of Cambodia to King Emmanuel, sank. The animals became very famous at that time in Europe, including a fable about battles between the rhinoceros and the elephant. Dürer drew the rhinoceros based on a sketch sent to him from Portugal.

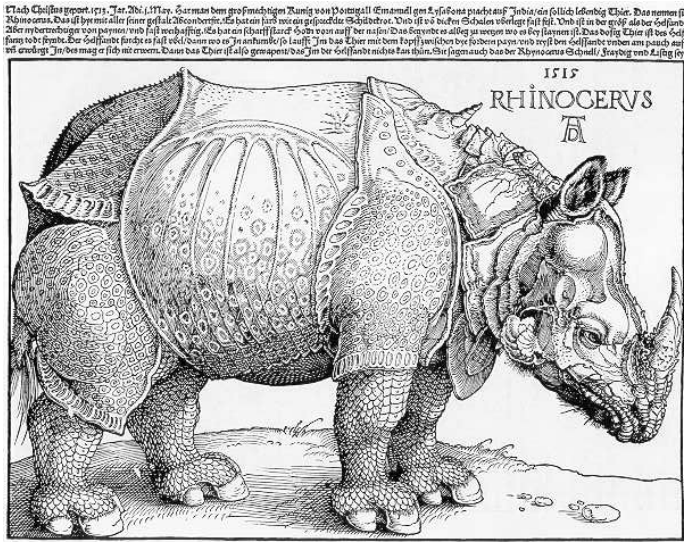


Figure 2.3: Engraving by Albrecht Dürer (1515). It was taken as an accurate representation of a rhinoceros for many years. Reproduced from Strauss,[101] page 509.

process. The big efforts have been concentrated on positional accuracy in maps. In addition, concepts like data quality and derived elements have been studied.[40,107,110] For instance, the concept of uncertainty as been understood as a multi-faceted characterisation of data. It can include confidence intervals, lineage, precision, standard deviation, heuristics, *etc.* In the end, uncertainty is a measure of doubt and distrust in the results.

There are three terms which are important to define. First, the error: it is the discrepancy between a given value and its true value. It is worth pointing out that in some situations the true value is simply unknown. For instance, information collected from the real world prior to any scientific modelling. At that level the maximum we usually expect is to replace a value of higher accuracy than the one we are assessing. A related term — accuracy — can be defined as indicating the closeness of results, or estimates to true values. Finally the term precision, with the connotation of similar results. As an illustration, note that the reading of temperatures can be precise (all the experiments delivering similar results) but not accurate.

Other authors have taken an interest in this topic, but focusing mainly on the visualization process. Craig Wittenbrink, Alex Pang, Suresh Lodha and colleagues have illustrated a number of ways to visualize errors made due to the visualization techniques used.[64,121] For instance, using vector glyphs to visualize uncertain winds and ocean

currents.[121]

Assuming that the uncertainty associated with data has been quantified, an important task is to map those metrics to visual cues. It must be said that visualization techniques used in different contexts, can be adjusted here, with the purpose of showing the accuracy of data. For instance, glyphs depict data through visual proprieties such as shape, dimension, size, orientation, *etc.* So accuracy can be conveyed by means of one of these visual cues. There are many other perceptive mechanisms that can be used: overlaying of primitives with the purpose of creating a balanced and unified pattern, maybe using transparency in order to make the image more readable; or the use of a side-by-side comparison of different solutions. Actually, similarity is a pre-requisite for the difference to be noted. Use of sound, blurring of image, animation, *etc.* , are other possibilities.

An important aspect is that errors at some stage of the visualization process may influence other visualization procedures further down the pipeline. For instance, error associated with raw data can imply discrepancy of results being magnified. In all, it is quite difficult to characterise the uncertainty of data as it is transformed through the visualization pipeline.

2.3.1 Variation of the Dataflow Model

As mentioned earlier, most of the MVEs are based on the dataflow model.[43] We can extend this reference model to accommodate the issue of accuracy.

Let us work with the variation of the Haber-McNabb model described by Brodlie.[10] In that approach, the input *raw data* is the sample of the underlying physical phenomenon to be visualized. There is a modelling stage where an estimate, or *model* of the underlying field is created. Then in the mapping stage, it is transformed to an abstract geometric representation. Finally the *geometry* is turned to *image* in a rendering stage.

Yet, the model itself may not be accurate; it is very likely that errors are introduced at the modelling stage — some due to the input data itself, some due to the model creation process (usually an interpolation). If instruments are used to collect the data then there

is always a degree of variability.

Figure 2.4 depicts the variation we envisage. It is desirable to have at the end of the modelling stage information about the accuracy of the model. We call it the *accuracy model*. Hence, *model* and *accuracy model* represent the underlying phenomenon better than the *model* alone. Moreover, we include *accuracy raw data* as input to the modelling stage, for instance to include metadata about the way the raw data is simulated or collected.

Next, the mapping stage where again errors are introduced. If we want to be able to describe the geometric representation of the underlying physical model, we have to provide the assumed *geometry* information plus a measure of its accuracy. For example, if the computed geometry is a sphere defined by its centre (x_c, y_c, z_c) and radius r, its accuracy can be given as a variation in the centre $(\Delta x_c, \Delta y_c, \Delta z_c)$ and in the radius Δr. Note that errors introduced in the previous stage will affect the accuracy of the mapping process.

Finally, the rendering stage, where the final *picture* embodies both the normal image and related accuracy. All the transformations that data have been subjected through the pipeline are reflected in the final picture. The goal is to show as faithfully as possible the underlying physical model we are trying to understand.

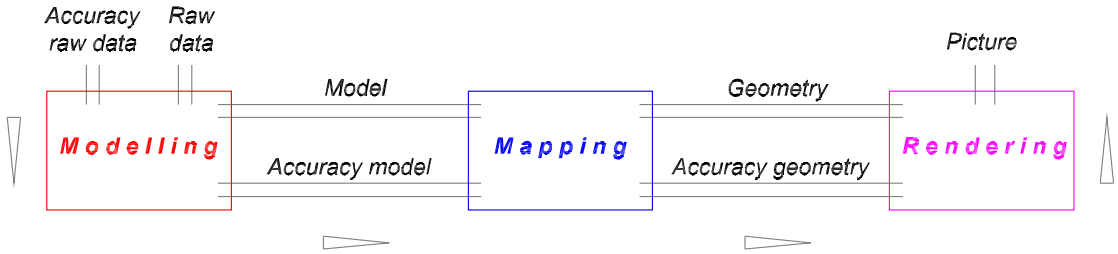


Figure 2.4: Variation of the dataflow model to accommodate accuracy information.

In the following Sections we will present some of the most common techniques used in scientific visualization.

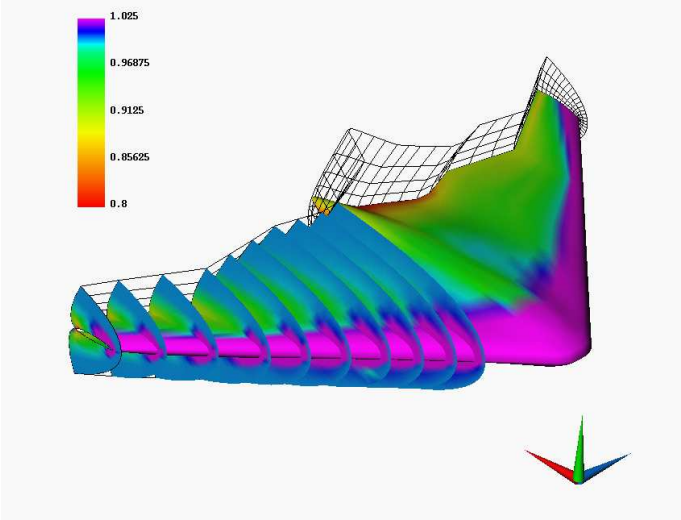


Figure 2.5: Visualization of pressure field over a Northrop B2 aircraft (near the airframe geometry). Adapted from an example found in IRIS Explorer.

2.4 Techniques for Scalar Data

These techniques aim to represent scalar data defined over a domain. The techniques we mention here are: graphs, bar charts and histograms, colour bands, surface displacement, contouring, surface rendering, and finally volume rendering.

There is one aspect of terminology that we would like to point out: in some literature such as reference 94, the term contouring is applied to techniques that depict scalar data of equalness, indiscriminately in two or three dimensional data sets. Throughout this work we make a distinction, keeping the traditional terminology: contouring is for two dimensional data sets whereas the term isosurfacing is applied in case of three dimensional data sets. We are aware that in the latter case some people use other terms such as surface fitting, surface extraction or surface tiling.

2.4.1 Graphs, Bar Charts and Histograms

These three constitute the set of elementary techniques to represent scalar data. A graph links data points over a continuous domain. Bar charts show data points defined over an enumerated set. It does so by means of bars whose length is mapped from the scalar data. The bars can be drawn horizontally or vertically. Finally, histograms

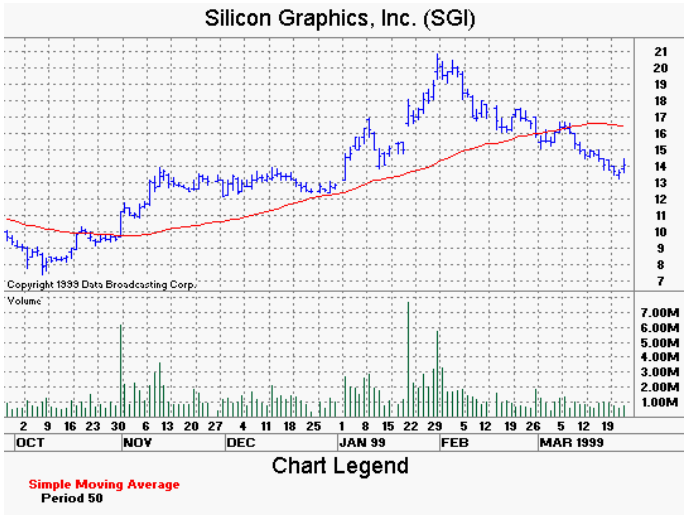


Figure 2.6: Charting the Silicon Graphics, Inc. security in the New York Stock Exchange. It shows the daily fluctuation of the share price (bars), accompanied by trading volume (bars) and moving average (graph). The moving average smooths the erratic short term variation (uncertainty) of the price line and effectively highlights when the price line has diverged from its average.

that show data aggregated into bins. That is, a data point is added to a bin if it lies within the corresponding range of values. Then bars of corresponding length are drawn.[11] Figure 2.6 shows an application of these techniques.

2.4.2 Colour Bands

The data is represented as image points, and coloured accordingly. Hence there is a transfer function between scalar values and colours. This function must be carefully chosen, otherwise the results can be very poor. The use of colour is an issue by itself. If one wants to enhance small variations, then for example the function can be algorithmic rather than linear. A common application is to create a lookup table of colours (discrete transfer function), where the scalar values are indices of the table. Figure 2.7 also shows this technique.

2.4.3 Surface Displacement

This technique aims to show scalar data of a two dimensional structured data set. The surface is displaced at each point based on the respective scalar value. Usually the

warping is normal to the flat surface, and may be affected by a scaling factor as well. An illustration can be found in Figure 2.7.

2.4.4 Contouring

This technique represents data of constant value defined over a two dimensional plane, by lines joining those points of equal value. It is a rather old technique that remains as useful as ever. Common examples are isobars in weather forecasting, and digital elevation data in maps. In Figure 2.7 there is an example. We discuss this technique in detail in Chapter Three.

2.4.5 Surface Rendering

Surface rendering aims to represent a surface drawn in the interior of a volume. The surface depicts points whose data values are equal. Hence the name *isosurface*. Actually this is the natural extension of contouring mentioned above.

There are many techniques for surface rendering. Here we will give an overview of these techniques. We make a major distinction between those that attempt to recover a surface from a set of two dimensional contours and those that work directly on volumetric data. One of the techniques, the Marching-Cubes, will be discussed in detail in Chapter Five.

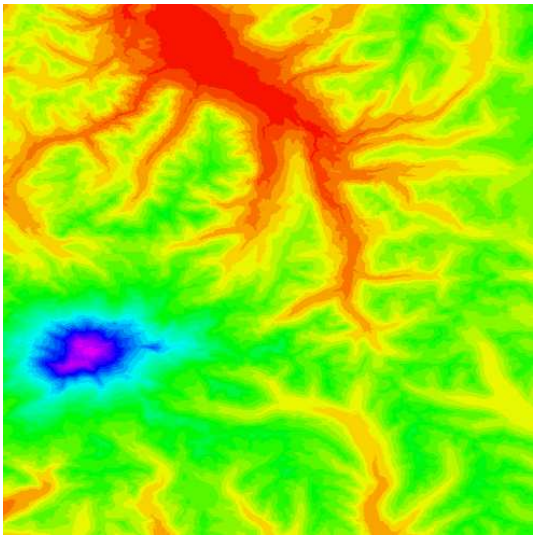
2.4.5.1 Surface Rendering from Planar Contours

In this pioneering technique the aim is to reconstruct the isosurface from contours of successive two dimensional slices of data. There are two major problems. They are:

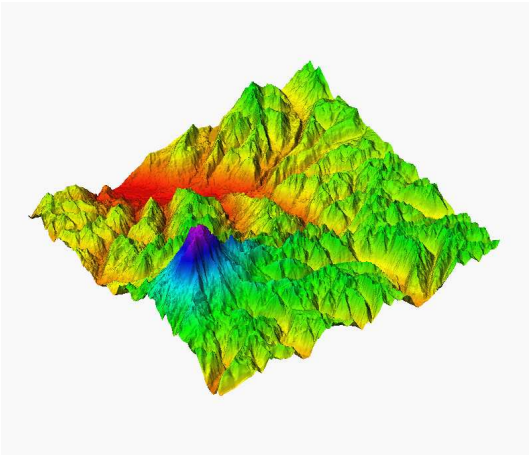
Correspondence.

To determine the topological adjacency relationships between contours from consecutive slices. In other words, which contours from one slice correspond to which ones from the other slice.

Colour bands.



Surface displacement.



Contouring.

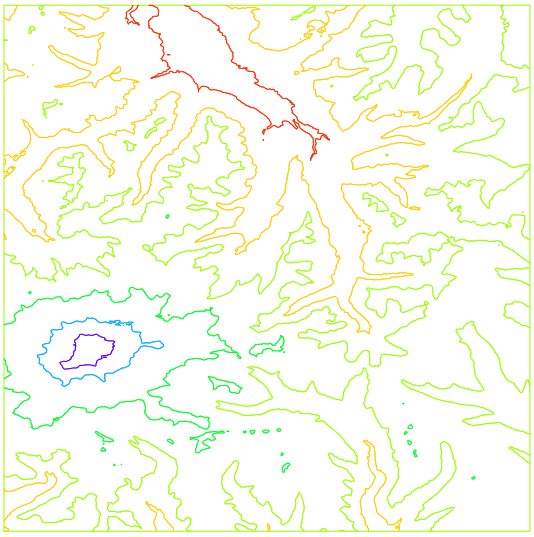


Figure 2.7: Three techniques to visualize scalar data defined over a plane: colourbands, surface displacement and contouring. As a note, this data set will be presented later in Section 3.4 on page 52.

Tiling and Branching.

To link correctly vertices of different contours in order to construct the triangular mesh, facing topologies such as those depicted in Figure 2.8.

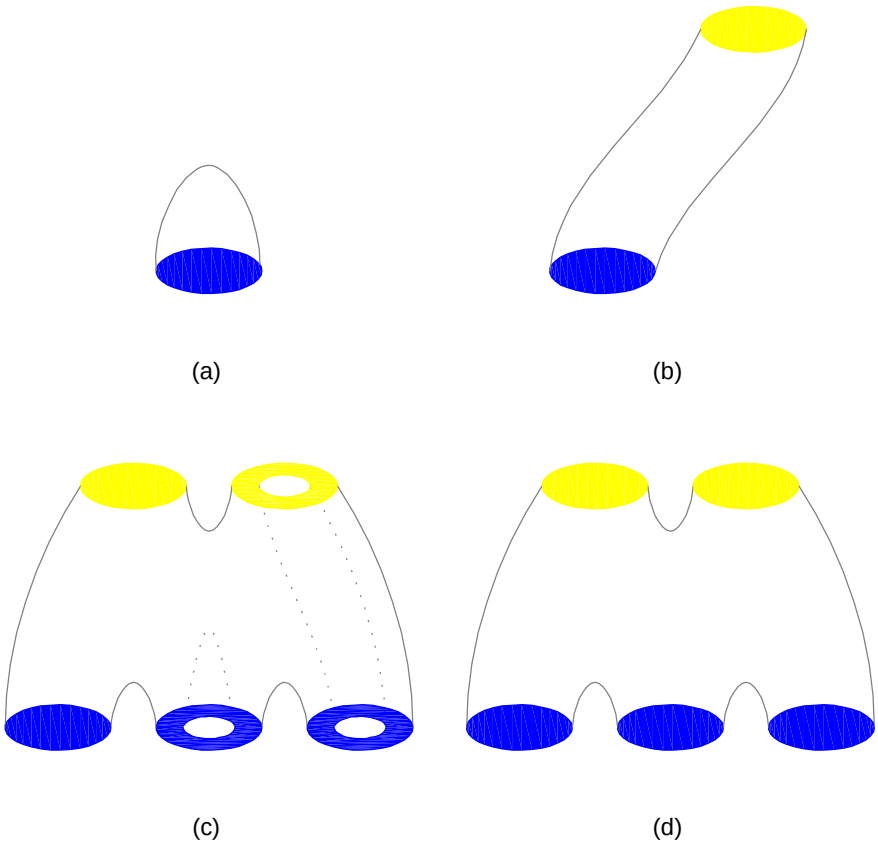


Figure 2.8: Possible connections in the reconstruction of isosurfaces from contours. On (a) there is no connection while on (b) there is a simple tiling between the two contours (blue and yellow). Situations of branching are represented on both (c) and (d), with and without holes respectively.

These problems are quite difficult to solve, particularly the definition of topology. Indeed, most of the techniques do not solve satisfactorily all the possibilities. There are two classes of algorithms: those so-called geometric algorithms that create directly an external triangulated surface, and those based on implicit functions. The latter are outnumbering the former.

A first technique to solve the tiling problem was proposed by Keppel.[59] In order to decide the orientation of a next tile (triangle), he use a directed graph representing

each contour, and heuristics metrics to find out a minimal cost path between contours. Fuchs *et al.* [37] adapted this idea but used a divide and conquer strategy instead of heuristics. These techniques only worked if there was a one to one relationship between contours, which frankly rules out the majority of real situations. Consecutive slices can show huge differences of topologies.

One solution is to join the contours together using a tiled strip and then tile the concatenation of the contours in one slice with the neighbouring slices.[72] Also, a tetrahedral mesh can be constructed using Delaunay triangulation to fill the volume between successive slices. Then the external parts are extracted. Another possible solution is to define an intermediate contour as the concatenation of the two original slices, which gives an easy correlation with data.[32] Or maybe an arbitrary number of extra intermediate cross-sectional contours.[82] But notice that the way interpolation is done is crucial.

It is very complicated to ensure topological correctness, at least automatically. The implicit based algorithms attempt to avoid topological considerations. The aim is to create a smooth implicit function, in which the sample data points fit. For example, Jones and Chen[54] developed a potential field function on the basis of the distance to each contour. The implicit function is then constructed by interpolating the field function of two consecutive slices. The resultant continuous volume space can then be visualized, whether using volume rendering or isosurface extraction, techniques outlined later.

2.4.5.2 Surface Rendering upon Volumetric Data Sets

Unlike the previous situation, these rendering techniques aim to extract the isosurface from a volumetric data set. Here we will indicate some of those.

Cuberille Approach. This technique presented by Herman and Liu (1979),[47] assumes that each voxel of data is of a constant value. Thus, the volume data is split into 0-voxels (outside the surface) and 1-voxels (inside). The surface is then represented by the faces of those voxels on the boundary between the two sets. It is a very simple method but the quality of the image is poor. Even with the help of shading to smoothen the image,[16,41] they still show the underlying voxel pattern.

Marching-Cubes. This is a widely used technique. The isosurface is found by examining the data values at the vertices of each data cell. A major feature of the method is a lookup table that lists how the surface can intersect with a cell. There are however some concerns regarding topological ambiguities. Since its introduction by Lorensen and Cline (1987),[67] this technique has been subject to a great deal of research. To name but a few, the asymptotic decider criterion that solves the ambiguities in the faces,[78] or the new methods outlined below: Dividing-Cubes, Marching-Tetrahedra and Discretized Marching-Cubes.

Dividing-Cubes. This method, suggested by Cline *et al.* ,[19] is a variation of Marching-Cubes. Each cube is divided into octants of the size of a pixel. This division is proportional to both image and data resolution, leading to the generation of the so-called internal pixel-sized voxels. These are then classified as inside, outside or on the surface, and displayed accordingly as points. The idea came about as result of reported situations where the data points in the cube were so close to each other that the triangles produced by Marching-Cubes were very small. By rendering points instead of triangles, the computational cost reduces significantly. However, zooming facilities are difficult to sustain since it would end up revealing the discrete representation of the surface — as dense point clouds.

Marching Tetrahedra. This technique emerged mostly as result of trying to solve the problem of ambiguity in the Marching-Cubes. The volumetric data is made of tetrahedra. The process is similar to Marching-Cubes but with a significant inferior number of possibilities for the intersection of the surface with the cell. A linear function is fit into each tetrahedron, yielding a representation of a triangle without any ambiguity. This method is resumed later on, in Section 4.1.1, when we discuss in detail ambiguity in Marching-Cubes.

Discretized Marching Cubes. In this technique, Montani *et al.* [73] suggested to merge both ideas of cuberille approach and Marching-Cubes. On the one hand, the binary array

of the cuberille method holds. On the other hand, the intersections with the cells in the boundary are established as with Marching-Cubes. But this time those intersections are defined as being halfway down each intersecting edge. The authors claim that the error introduced with this simplification is affordable in many practical situations.

Particle Systems. In this approach the extraction of isosurfaces relies on the dynamic behaviour of particle systems. First, particles are released between any adjacent sample points of the data set whose values are on opposite sides of the threshold value. Then they are attracted towards the isosurface while simultaneously repelling adjacent particles. A birth-death process is developed and once the equilibrium is reached the positions of particles are used used as vertices of the surface to be rendered. As presented by Crossno and Angel,[22] this method also allows control of the density of the mesh vertices on the basis of isosurface features such as curvature.

Direct Surface Rendering. This method, as works by Jones and Chen[55] and Parker *et al.* [84] show, uses the ray tracing technique to obtain the isosurface so avoiding the creation of an explicit representation for the isosurface. The technique is as follows: from each pixel on the screen, rays are cast through the volume data. Whenever the isosurface is intersected by those rays the computed intersection point is rendered. Usually shading techniques are included in the process. Notice that if one wants to render a mesh one can always create a volume data using voxelisation and then use direct surface rendering.[53]

2.4.6 Volume Rendering

Volume rendering maps directly the volumetric data set into the display, enabling greater data understanding by laying emphasis on the overall as opposed to the detail. There is no intermediate geometric representation of the data. The underlying idea is to see through the data set which is considered as a light-participating medium. Concepts of lighting and shadow are crucial to depict the interior of the data set. The major drawback though is that it is very costly. Furthermore, each time the image is rendered (for example

if the user wants to see from a different viewpoint) the entire data set has to be traversed again.

The volume rendering techniques are broadly classed as either object order or image order, although hybrid methods can be set as well. The major difference is that in image order algorithms, a ray is cast from each pixel of the image plane onto the data set (backward projection), whereas in object order, the volume is traversed cell by cell and each one is projected onto the image plane (forward projection). In the following we present these strategies in general. The emphasis is on techniques for regular data sets. In the end these basic techniques have been adjusted for irregular data.

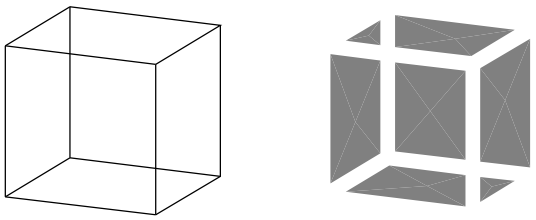
Reviews on the subject can be found in references 33 and 89.

2.4.6.1 Object Order Approach

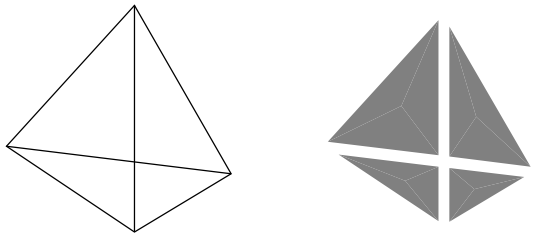
The first technique is based on concepts of rendering polygons in a scene. Indeed, the projection of a cell into the image plane can be regarded as a set of regions of pixels, where each region is a two dimensional polygon. Figure 2.9 shows the projections to consider; in case of a parallelepiped cell there are up to seven polygons, which can be triangles or quadrilaterals. With tetrahedra, there are up to four triangles.

Thus, the shading model uses the colour and opacity assigned to each vertex of the polygon and interpolates them for the remaining polygonal area. It is basically a Phong model. Those values of colour and opacity are then merged with information already collected from previous cells. This rendering approach was used by Drebin *et al.* (1988),[26] who first transformed the volume in order to be perpendicular to the viewing angle, and then used a scan line method. Upson and Keller (1988),[109] on the other hand, used a z-buffer technique.

In a different approach, Westover (1990)[119] projected each cell onto a point of the image plane but the pixels around that point were also affected by the colour and opacity of the cell. The degree of influence is defined by a reconstruction filter. This filter, normally a gaussian, is called the *footprint*. The influence on neighbour pixels decreases outwards from the projected point, as in the image obtained as result of throwing a snowball on



(a)



(b)

Figure 2.9: Projection of cells into the image plane in volume rendering, considering the object order approach. A cell is perceived as a set of two dimensional polygons. On (a) the cube is projected to seven polygons while on (b) the tetrahedron is projected to four polygons.

a wall. Hence the name *splatting* for this popular technique. All the contributions are composed together to create the final image .

2.4.6.2 Image Order Approach

This approach is based on the classic ray casting; rays are cast from the image plane into the data set. The main difference is that in the classic ray casting only the first intersection of the ray counts whereas here, opacity of all elements are taken into consideration (see reference 34 for details on generic ray casting).

The general scheme, based on the model of Blinn (1982)[5] to model clouds and dusty surfaces, and then used by Levoy (1988),[62] is as follows: as the ray passes through the volume, data values are sampled at a convenient sampling rate (between entry and exit points). The sampling is normally done using trilinear interpolation. Then a process, the so-called *compositing*, takes the colours and opacities of the sample points — which

are defined by the scalar value — and composes them into the final value for the pixel. In front-to-back compositing, the opacity of points close to the image plane will determine the degree of contribution from more remote points. If, for instance, one is opaque, then contributions from further points are discarded. The transfer function that determines the final colour C of the pixel is

$$C = \sum_{i=1}^{n} c_i \, \alpha_i \prod_{j=1}^{i-1} (1 - \alpha_j) \, , \tag{2.1}$$

where n is the number of sample points along the ray, and c_i and α_i the colour and opacity of the sample point i respectively.

The above transfer function models light passing through a number of adjacent, semi-transparent gels. A gel is a transparent medium in which a large number of opaque spherical particles of fixed radius, non-uniform distribution and varying reflectance are suspended. As the number of particles in the gel increases, less light will pass through it; the degree of opacity of the gel is related to the number of opaque particles it contains. In (2.1) the opacity ranges from 0 (totally transparent) to 1 (opaque).

There is also a popular but more complex method elaborated by Sabella (1988).[91] He considers the data set as a varying light emitter. It is similar to the system of particle light sources of Reeves,[87] but it models the density of particles in a region, not the particles themselves. Unlike the previous model, the intensity of light that reaches the eye allows continuous variation of density.

2.5 Techniques for Vector Data

The aim of techniques for vector data is to represent in three dimensional space the direction and magnitude of each data value. One important form of application is the visualization of flows. In that respect it is worth mentioning two review papers, by Post and van Walson[86] and by Delmarcelle and Hesselink[29] respectively. Here we outline local icons, particle tracing, streamribbons, streamsurfaces and streamtubes, the topological approach and finally techniques based on texture.

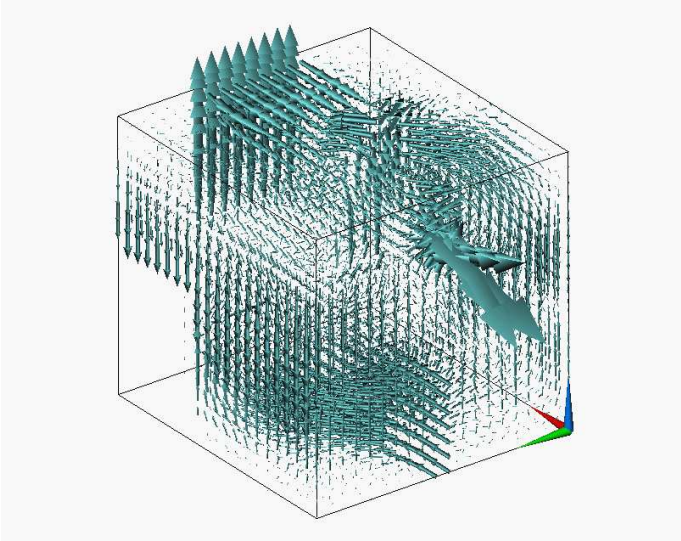


Figure 2.10: Visualization of a velocity field using arrow plots as icons. The data is the office data set available in reference 94.

2.5.1 Local Icons

This is a straightforward technique where geometric entities are drawn at each data point. They are named icons or glyphs. The dimension, shape, orientation and colour are related to the vector data at that point. The dimension (length, area, volume) can be balanced to control the clutter of the final image.

The simplest icon is an oriented line, sometimes referred to as *hedgedog*. The oriented line can have an arrow at the end, hence the name *arrow plots*. Figure 2.10 shows an example of an application of arrow plots.

The geometry of glyphs can be extended to higher dimensions. For example triangles, cones or cuboids. However we would like to stress the fact that scaling in this case can be misleading. For instance, if two different values are mapped into squares, the difference of areas between the squares is magnified by the square of the scale factor. The problem is the visual perception is mostly directed towards the area of each square.

The major problem of glyphs is that position and orientation are in general difficult to perceive in space.

2.5.2 Particle Tracing

The concept of this technique derives mainly from experimental work. In order to provide a framework we present the following definitions:

Pathline.

Line traversed by a particle, during some interval of time.

Timeline.

Line formed by linking a series of particles released at one instant. Due to the flow, particles individually move after a time interval to a new location, through which the form and location of the timeline alters.

Streakline.

Line composed of particles which have passed through a specific location during a certain time interval.

In addition, another line representation can be derived mathematically: the *streamline*, a line that is at all points tangent to the instantaneous velocity field. In case of steady flows (not dependent on time), pathline, streakline and streamline coincide.

Therefore, particle tracing aims to show the various positions of a particle moving in the field. Since the velocity of the motion is by definition the derivative of the position in relation to time, then the solution of the problem resides in integrating numerically the vector field, step by step. A good review of particle tracing can be found in the reference 86. We discuss this technique in detail in Chapter Six.

Kenwright and Mallinson[58] suggested however a different approach: a streamline is conceived as the line of intersection between two surfaces that are at all points tangent to the flow field (called streamsurfaces outlined below). These surfaces are defined as implicit surfaces. The feature of this technique is that it follows the law of mass conservation, and does not require time stepping.

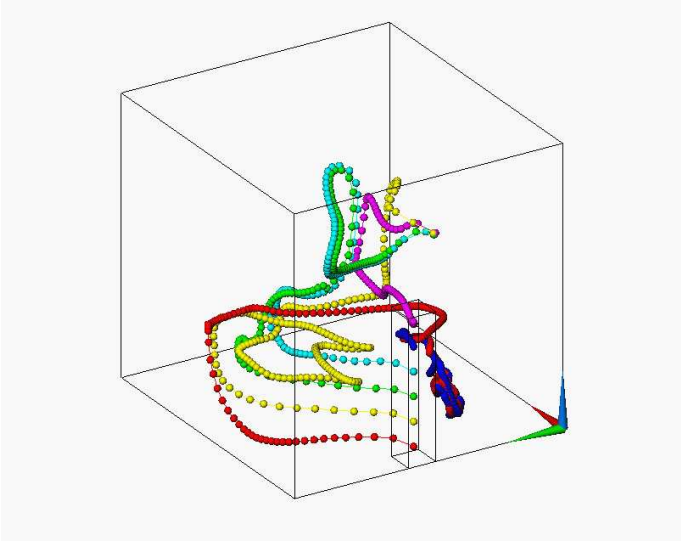


Figure 2.11: Particle tracing. The data is the office data set available in reference 94.

2.5.3 Streamribbons, Streamsurfaces and Sreamtubes

This class of methods is a natural extension of streamlines. On the first level, two adjacent streamlines can be bridged with a polygonal mesh to form a so-called *ribbon*. Take notice that the streamlines should remain close enough since a ribbon should be tangent to the field at all points. With this technique it is possible to show the rotation of the flow (vorticity) and the spread of the flow (divergence). Indeed, the changing width of the streamribbon is related to the cross-flow divergence of the flow whereas the amount of twisting indicates the vorticity.

By definition, a streamsurface is a collection of a infinite number of streamlines passing through a base curve, called a *rake*. Streamtubes are obtained if the rake is closed. One method to generate streamsurfaces is to advect particles from the rake and then create a polygonal mesh that connects adjacent streamlines (as for streamribbons). In many cases the streamlines diverge so that the resulting surface is poorly defined. In order to avoid this, Hultquist[49] used an advanced front of a steady growing streamsurface. Particle traces and tiling is done concurrently. The number of particles and spacing between them is adjusted at the front in order to keep the distance between particles constant. Also, van Wijk[115] represented a streamsurface as an implicit surface, showing the sweep of an initial curve through the field. The shape of the initial curve is defined

by the value of the implicit curve at the inflow boundaries. A streamsurface can also be represented by means of surface-particles, a texture based technique to be mentioned later.

Finally we would like to cite the concept of streamballs introduced by Brill *et al.* [7] This method is based upon implicit surface generation adopted from metaballs. They have the ability to automatically split or merge with each other, depending on their distances. By advancing appropriate skeletons through the field, and displaying the resulting streamballs, one can obtain various techniques, such as streamsurfaces.

2.5.4 Topological Approach

One important aspect in vector fields is the understanding of its inherent structure. This is mainly achieved by studying the critical points, points at which the magnitude of the vector field vanishes, thus leaving the direction undefined. Either the vector field converges or diverges and/or circulates around a critical point. This behaviour is determined by the eigenvalues of the Jacobian, the matrix of partial derivatives. The imaginary part indicates rotation whereas the real part the relative attraction of repulsion. Critical point theory is then applied here. Take notice that it is not so easy to perceive this type of analysis of global features.

Helman and Hesselink[46] proposed to form a 2D skeleton by connecting the critical points on the surface. The field is then divided into separate regions topologically equivalent to uniform flows. Next, streamlines are generated starting from points on the skeleton. These allow the construction of stream surfaces, with adaptative refinement in areas of divergence.

2.5.5 Texture Synthesis Based

This class of techniques are based on the idea of having moving particles to create the illusion of fluid movement. By overlapping particles, a stochastic texture is obtained. This is a throwback to the particle systems introduced by Reeves to model fuzzy objects such as fire or clouds. There, the particles have a lifetime cycle of birth, life and death. While

alive, the various attributes such as position, speed, and direction of motion, vary in time according to stochastic models.[87,88] (actually Blinn[5] used a similar model before, see volume rendering on Section 2.4.6). One famous example of the application of particle systems was the animation of the atmosphere of Jupiter in the film ”2010” .[127]

This class of methods are generally costly. But one has to acknowledge their ability to provide pictures that really show how the flow field looks like, in some cases denoting proper experimental work.

In the following we present the approaches: extension of volume rendering, surface-particles, spot-noise, line integral convolution and motion maps.

Extension of Volume Rendering. Although volume rendering is traditionally a technique to visualise 3D scalar data, the concept can be extended to vector data by adding texture that in some way depicts the vector field. For instance, texture can be added to a contour surface of cloud density and advected according the wind field, as Max *et al.* [70] did. The point is to use a 3D volume texture function evaluated on a contour surface.

A different approach was taken by Ma and Smith,[68] using what they referred to as *virtual smoke*. After defining a seed point, the volume surrounding it grows voxel by voxel, based on the vector field near the seed point. The image is thus updated in order to animate the growth of the seed.

In another example, by Crawfis and Max,[21] the basic splatting technique (see Section 2.4.6.1) was used. The vector field was integrated into the scalar reconstruction function and so splats were textured to depict the vector field. The direction of the polygon splat shows the direction of the flow, and its final colour can also convey the magnitude of the vector field.

Surface-particles. By adding spatial information to the concept of particle systems, van Wijk modelled a particle as a very small surface. Due to its size, the shape is irrelevant and so a particle is modelled as a point with a normal. Adding orientation information is conducive to a more advanced shading model. If using a large collection of surface-particles, one can represent a (stream) surface moving with and deformed by the vector

field without having to compute it explicitly. Particles are released in space and time, and then advected.[100, 113, 114] With animated surface-particles, one obtains a very good view of flow field.

Spot Noise. This technique, introduced by van Wijk,[112] draw spots of random size and intensity in a plane, giving a texture effect. The texture is defined as:

$$f(x) = \sum_i a_i \, h(\, x - x_i \,) \, , \tag{2.2}$$

where x_i is the random position, a_i the random scale, and h is the spot function — zero everywhere except for an area that is small compared to the texture size.

Different spot shapes and size yield different textures so the global appearance can be controlled. Moreover, the shape can be adapted to the data at that position, by scaling the spot in the direction of the vector field proportionally to its magnitude, for instance. However, when data varies rapidly, the results are not satisfactory. More sophisticated spot shapes such as bending the spot in the direction of a stream surface were introduced.[25]

Linear Integral Convolution. This method presented by Cabral and Leedom[14] is a powerful technique to visualize and animate vector data in a plane. It is based on a common technique in image processing: image convolution. The texture is generated by convolution of an input texture with a one-dimensional filter kernel. In its simple form, the convolution filter is a straight line tangential to the local vector. A final pixel $O(i, \, j)$ is obtained by the weighted sum of a number of pixels along the line in the input texture:

$$O(i \, , j) = \sum_{p \, \in \, \Gamma} I(p) \, h(p) \, , \tag{2.3}$$

where Γ is the set of pixels in the input texture, $I(p)$ is the input texture pixel at grid cell, and $h(p)$ the convolution filter. In effect, the input texture is blurred as an image tangential to the vector field, at pixel-resolution. This correlates the resulting pixel values along streamlines, so that a sense of flow direction is obtained. Figure 2.12 gives an example.

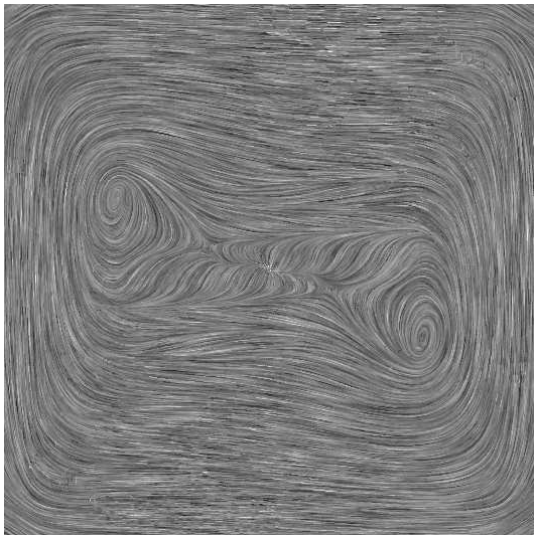


Figure 2.12: Line Integral Convolution. Visualization of a 2D horizontal slice of a flow velocity data set. This data set is the double glazing data set that will be studied in Chapter Five.

Later developments include: implementing on curvilinear grid surfaces[36] and unsteady flows;[35,98] separating the streamline integration from convolution, making the algorithm resolution independent;[99] modifying the frequency of the input noise to gather extra information such as the magnitude of velocity, in the image;[60] encoding orientation of the flow;[118] and finally using in 3D flow volumes.[50]

Motion Maps. This technique is used in two dimensional steady flow fields. The basic idea is to have a dense coverage of streamlines that encode flow information by means of colour, and then using colour table animation techniques.[52]

The image is perceived as a grid of cells covering the flow field. Each pixel is assigned to an index (entry) in the colour table. By cycling the indices, the image gives a sense of motion. If consecutive pixels are assigned to the same index then a different speed is shown. Correlation between neighbour pixels is therefore important for the animation.

2.6 Techniques for Tensor Data

Tensor data consists of nine scalar functions of position, which should be visualized as a group. They are partial derivatives of the vector field, aiming to show its distortion. In general, there is a decomposition into a symmetrical part — stress-strain — and an anti-symmetrical one which denotes rotation. The eigenvectors are called the principal direction of the tensor, and along these vectors the deformation is pure stretching. Quite understandably, tensor visualization is rather difficult to get a mental picture of. In the following we introduce some techniques for tensor visualization, classed as tensor glyphs and hyperstreamlines.

2.6.1 Tensor Glyphs

Tensor glyphs are icons designed to show the eigenvectors. The most primitive will be an ellipsoid with principal axes as the three eigenvectors.

Haber and McNabb[43] used a glyph consisting of a shaft and a disk to visualize a symmetrical stress tensor. The shaft was oriented according to the direction of largest stretching, whereas the elliptical disk was aligned with the other principal directions. Also, de Leeuw and van Wijk[24] developed a more advanced glyph to visualize the Jacobian of a velocity field (spatial derivatives of each velocity component). First, the tensor is decomposed into parallel and perpendicular components to the flow. Then the following components are singled out: velocity, curvature and rotation, acceleration, shear, and convergence and/or divergence.

Again, as in visualization of vector data, multiple instances of glyphs are released in the field, which does not avoid a loss of continuity. Furthermore, cluttering of the image is a serious problem. Even more so with tensor glyphs than with vector glyphs.

Another technique that can be included in this class is the *stream polygon*, introduced by Schroeder *et al.* [93] Here, glyphs as regular n-sided polygons perpendicular to the local vector field are placed along a streamline. The shape is distorted according to the local deformation in the field. Effects like the twisting or scalar parameters of the field are displayed by accordingly rotating and shearing the polygons or changing attributes

like radius or color.

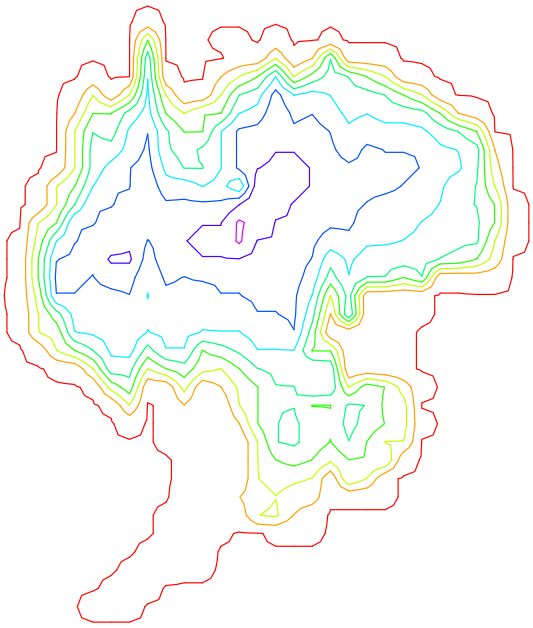
2.6.2 Hyperstreamlines

Hyperstreamlines are the extension of streamlines used in vector data, allowing to show real, symmetrical tensor fields along a trajectory, continuously. Delmarcelle and Hesselink[27,28] produced hyperstreamlines as follows: a primitive geometry, usually an ellipse, sweeps along one of the eigenvectors while stretched out in the traverse plane. The stretching is affected by the other two orthogonal eigenvectors. Then all the generated primitives are linked along the trajectory. Colour can be used in respect to the eigenvalues, usually representing the longitudinal eigenvalue. With an ellipse, a tubular shape is obtained. Another possibility is a cross that generates a helical shape. The selection of one or the other depends on the type of analysis of interest.

Chapter

THREE

Contouring



There is no doubt that contouring is still a major application in computer graphics. Indeed, it is widely used in cartography, weather forecasting, fluid dynamics, and in many other scientific fields.

The layout of the chapter is as follows: first, we provide an overview mentioning the roots of contouring as well as the general methodologies. Then the focus is directed

towards a very common class of contouring, when data is defined pointwise over a rectangular grid and assumed to follow a bilinear interpolant on each cell of the grid. Contouring methods construct a piecewise linear approximation to the contours of the bilinear interpolant. The current state-of-art is presented, with two methods in particular being described, and then our new approach is introduced. Two data tests are included to highlight the virtues and pitfalls of each method. Finally we consider the representation of accuracy in contouring, targeting the new method.

One key concept throughout this chapter is that of correct topology. We consider a solution as topologically correct if it shows the same topology as that of the underlying data interpolant.

3.1 Overview

The general concept behind contouring is that of isolines (lines of equalness): a set of lines that show by their absolute and relative positions the locations and gradients within a set of numbers.[90] These numbers can be temperatures, elevations from the ground, population densities, *etc.* The isolines can also be named according to the type of data involved: isotherms (temperature), isobar (pressure), *etc.* In case of geographic elevation data, cartographers call them contour lines, an expression which is sometimes used interchangeably with isolines. Since our main interest in contouring is just from the point of view of visualization technique, hereafter we follow the general approach of using both terms isolines and contour lines.

There are several methods to generate contour lines, for different types of grids, interpolations, and orders of curve generation. For detailed information on contouring one can read the references Brodlie,[8] Sabin[92] and Watson.[117]

3.1.1 Roots of Contouring

To find the roots of computer contouring one has to go back to the sixteenth century. At that time the manual preparation of maps to help sailors was a high-profile task. Questions such as ”how deep is the water” or ”which way is north” were better answered

referring to pictures. So contour lines were drawn on topographic maps to represent elevations above sea level, and on nautical charts to represent depths. Many examples of contour maps produced at that time can be found. Robinson[90] mentions that the first contour map is credited to the Dutch surveyor Pieter Bruinss who in 1584 showed the depth of water of the River Spaarne (isobath map). But it was not until 1777 that a Frenchman, Meusnier, first published contouring as a technique to show surface configuration from elevation data.[90] As far as computer contouring is concerned it appears that the suggestion by John von Neumann in the 1940s to interpolate automatically meteorological data triggered the events.[83]

3.1.2 Methodologies

Most of the contouring techniques apply to gridded data which means data is prepared before the contouring itself takes place. In the previous Chapter in Section 2.2, we have presented the main types of data grids. In 2D, the common grids are characterised by rectangular cells, or else triangular cells.

There are two main strategies to extract contour lines from gridded data: by cell sequence or by contour sequence.

Cell sequence. Each grid cell is scanned and examined, one by one. The basic assumption is that the contour passes through the cell in a finite number of ways. If grid values higher and lower than the contour value are found, then part of the contour is within the cell. In that case, intersection points with the edges of the cell are computed using some type of interpolation. Then these points are connected by line segments or a smooth curve is fitted through them. In the latter case the smoothness of the curve can be controlled using a variable step size and/or information such as the gradient.

Contour sequence. An initial pass is made through the data to identify the grid intersections with a contour line. One of these points is chosen as start point, and the contour is tracked through the grid, noting the intersection points which have been traversed. A remaining intersection point is chosen as the next contour start point, and the process

continues until all the intersection points have been used up. As a note, with this strategy it is very easy to place labels at regular intervals along the contour, which is useful in some cases.

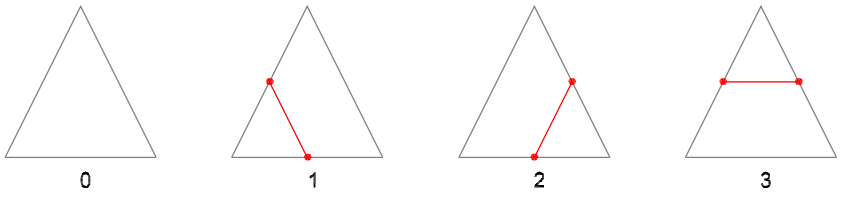
Whether using cell sequence or contour sequence strategy, the order of the data and the type of grid where it is defined will determine the appropriate choice of technique.

Linearity between adjacent data points. Linearity yields the simpler methods. If the grid is triangular the interpolant function is assumed to be linear in the interior and so the contours are straight lines. So it is just a matter of finding the intersection points with the triangle edges and connect them. In case of rectangular cells, the interpolant is bilinear so a conic curve should be drawn for a contour line. Figure 3.1 shows the cell classification for both grids. Bilinear data over rectangular grids is going to be the main emphasis of this chapter.

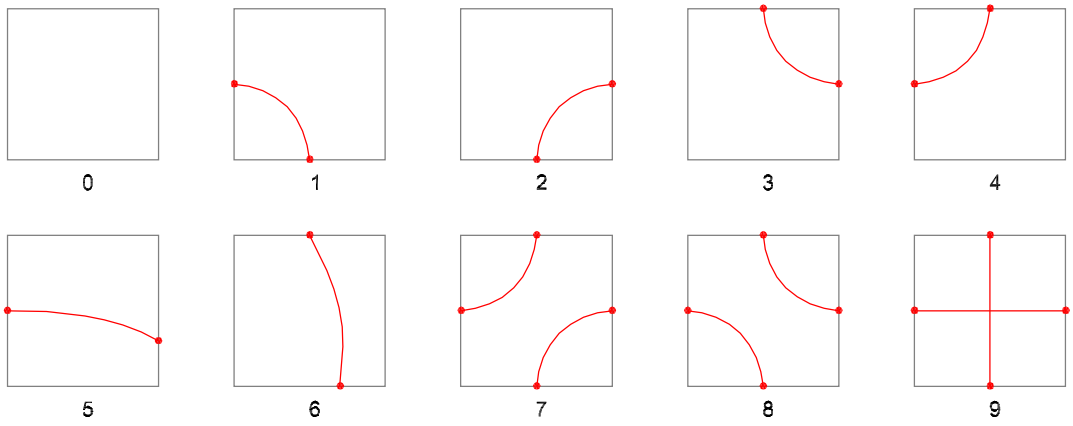
Non-linearity between adjacent data points. Unlike data varying linearly between adjacent data points, intersections with the edges may not be unique, as shown in Figure 3.2. Many techniques can be used, though not simple. In general the stepping used to compute successive line segments to approximate the contour is increased. Another option is to refine the grid in such a way that techniques suitable for data varying linearly over cell edges can still be applied. This requires extra memory but it is particularly beneficial if there are multiple contours to draw.

3.2 Bilinear Data on Rectangular Grids

Data varying linearly between adjacent data points is a very common assumption in contouring. As mentioned before, even if data varies non-linearly a higher data resolution obtained in a pre-processing stage can enable the visualization techniques to work on the assumption of linearity.



(a)



(b)

Figure 3.1: Cell classification for (a) triangular and (b) quadrilateral data cells, when data varies linearly over the edges of the cells.

Formally the problem is as follows: given the values of a bilinear function $F(x, y)$ at the vertices of a rectangular cell D, compute and display isolines of threshold value Θ

$$C_\Theta = \{\, (x,\, y) \mid F(x,\, y) = \Theta\,, \quad (x,\, y) \in D \,\}\ . \tag{3.1}$$

For convenience and without loss of generality we transform the cell domain D into a unit square. Hereafter we shall assume the bilinear function to be defined within a unit square $[\,0, 1\,] \times [\,0, 1\,]$. Therefore

$$F(x, y) = a\,x + b\,y + c\,x\,y + d\,, \tag{3.2}$$

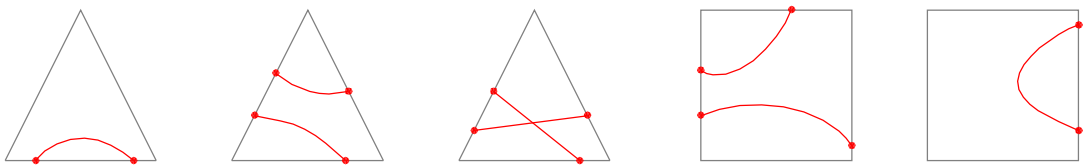
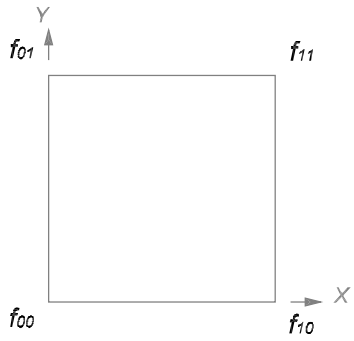


Figure 3.2: Non-linearity between adjacent data points. Contours may intersect cell edges at more than one location, regardless of the type of grid (triangular or rectangular).

with



$$a = f_{10} - f_{00}\,,$$
$$b = f_{01} - f_{00}\,,$$
$$c = f_{00} + f_{11} - f_{01} - f_{10}\,,$$
$$d = f_{00}\,,$$

where f_{00}, f_{01}, f_{10} and f_{11} are values of $F(x, y)$ at cell vertices.

The partial derivatives of (3.2) are

$$F_x = a + c\,y\,, \qquad\qquad F_y = b + c\,x\,. \tag{3.3}$$

The point $S = (-b/c\,, -a/c)$ is a stationary point since it is the point where the two derivatives F_x and F_y vanish. The value of F at the stationary point is

$$F(x_S,\, y_S) = \frac{f_{00}\, f_{11} - f_{01}\, f_{10}}{(f_{00} + f_{11}) - (f_{01} + f_{10})}\,. \tag{3.4}$$

The eigenvalues of the Hessian matrix are of opposite sign, $\lambda = \pm c$, thus the stationary point is a saddle point. This point S is defined as long as $c \neq 0$. But if $c = 0$, F is linear and so contours are just straight lines. Contouring in that case is trivial and more importantly exact.

Assuming $c \neq 0$ the contour curve within the cell is a hyperbola. The problem is then reduced to drawing the hyperbola within the limits of the cell knowing F. The

possible configurations to depict the contour curve were shown before in Figure 3.1(b). The hyperbola representation requires some kind of approximation. Since execution time is usually a key factor, one can (roughly) approximate the hyperbola by straight lines connecting the intersection points with the cell edges. That is the basic method.

For the sake of clarity let us introduce the following concepts about cell points and degenerate contours. The configuration 9 of Figure 3.1(b) shows a degenerate contour as we conceive it.

Definition 3.2.0.1 (Cell points). *A cell point* (x, y) *is positive (negative) when its data value is above (below) the threshold value* Θ*. We define intersection points as points in the cell edges cut by the contour. In the particular case of cell vertices being intersection points we refer to them as intersection vertices* [a]*.*

Definition 3.2.0.2 (Degenerate contour). *Given the bilinear function* F *of* (3.2) *the corresponding contour is said to be degenerate if it is described topologically as two straight lines.*

3.2.1 Ambiguity

The configurations 7 and 8 of Figure 3.1(b) show the hyperbola intersecting all the four edges of the cell. That is the case when positive and negative vertices are diagonally opposed. The pairwise connection of intersection points is ambiguous since it could be in up-right or up-left direction, from the bottom point. Any solution to solve this ambiguity should provide a consistent approach ensuring positional continuity between adjacent cells. But the final aim is really to obtain a solution which is topologically correct in terms of the bilinear interpolant and as accurate as possible.

[a] As a curiosity some authors in the past have suggested to add to each data value a small random displacement in order to prevent these cases. They argued that computers have a finite representation anyway so in the end the added distortion was not so significant. Definitely we do not accept this idea.

Four triangles technique

According to Sutcliffe,[102] this method has its roots in the work of Dayhoff (1963) and Heap (1974). Heap used the idea of Dayhoff to approximate the value at the centre of the rectangular cell by averaging the values at the four corner points, and then dividing the cell into four triangles. Hence we name it as the four triangles technique. Having done that one can obtain two extra points lying on a diagonal of the cell, using linear interpolation. Figure 3.3(a) shows an example for a threshold value $\Theta = 0$.

There are a few aspects that ought to be mentioned. First, the averaging process corresponds to using a bilinear interpolation. This interpolation model is contradicted by using a linear interpolant within each triangle. If one assumes that a linear interpolant was correctly applicable within each triangle then the contours are straight lines, so the solution is exact. But that premise is not valid in our case. Second, the method provides consistency of solutions from cell to cell. Third, unfortunately it does not always provide a correct topology as Figure 3.3(b) depicts. It is only correct when both the saddle point and the centre of the cell lie in the same region of the hyperbola. In other words, when the function values at saddle point and centre of the cell have both the same sign.

Asymptotic Decider

This method was introduced by Nielson and Hamman (1991)[78] to solve ambiguity in the Marching-Cubes method, an isosurfacing technique discussed next chapter. It is based on the following criterion: the pairwise connection is established so as to cut off the vertices of opposite sign to the saddle point value. Figure 3.4 depicts the method. This method provides always a correct topology for the bilinear interpolant so consistency as well. Recall that the saddle point is the intersection of the asymptotes of the hyperbola ($F_x = 0$ and $F_y = 0$). Those asymptotes will indicate the correct topology — there is no crossing between the hyperbola and the asymptotes.

From now on we shall assume that the basic method applies this decider criterion to resolve the ambiguity.

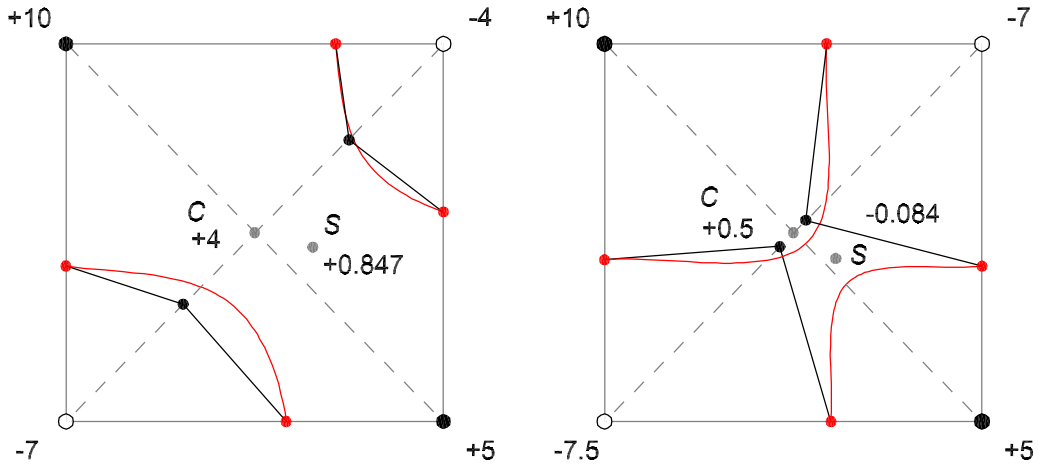


Figure 3.3: The four triangles method, with a threshold value $\Theta = 0$. It provides a topologically correct solution on (a) — the function values at the centre of cell C and at the saddle point S have the same sign — but not in the case of (b) where those values have different sign. In the latter case the two extra points are assumed to lie on the wrong diagonal.

3.3 Proposed Contouring Method

We now aim to improve the basic contouring method, following the cell sequence strategy. As above, we shall assume linear data defined over a rectangular grid. Without loss of generality the cells are units, *i.e.* defined over $[\,0,\,1\,] \times [\,0\,,1\,]$ and the threshold value Θ is zero. The goals are:

1. To ensure topological correctness as defined in the introductory section. In this case positional continuity between adjacent cells is also ensured.

2. To generate contours more accurately than the basic method.

3. To have an acceptable computational cost.

For each cell the method works as follows: first we check whether there are vertices intersected by the contour or not. If there are then the (particular) situation is dealt with accordingly. Otherwise we compute the intersection points on the edges. If the interpolant is linear or the contour is degenerate then the representation is trivial: straight line(s). If

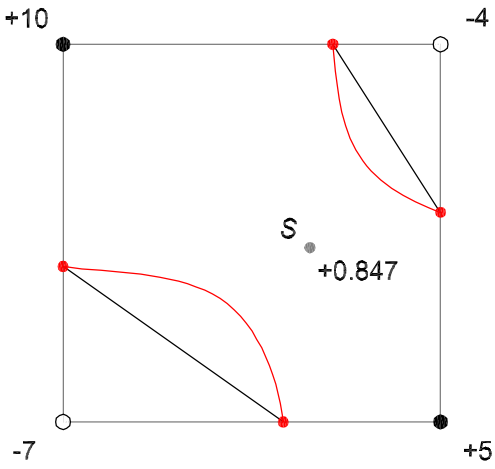


Figure 3.4: The basic method and the asymptotic decider criterion to solve ambiguity. In the example above for a threshold value $\Theta = 0$, the function value F in the region between the two contour sections is positive, which includes the saddle point S. Notice that S is the intersection of the two asymptotes $F_x = 0$ and $F_y = 0$.

not the next step is to solve any ambiguity; in case of four intersection points we use the asymptotic decider as described in Section 3.2.1 to establish the correct topology. Once the pairwise connections are set, we include an extra point for each pair of intersection points. Each contour section is drawn as a polyline from one intersection point to the extra point and then to the second intersection point. Next we will discuss how to obtain this extra point. We call it a *shoulder point*.

3.3.1 Shoulder Point

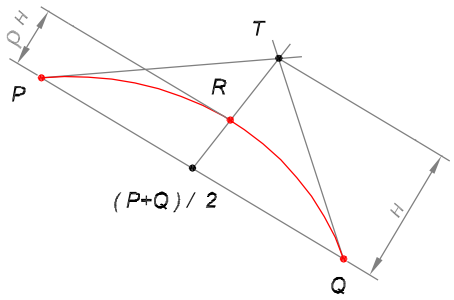
Looking to the classical manual method of drawing a conic curve, the problem is: ”given two end points and corresponding tangent lines, and an intermediate point, find a set of intermediate points sufficient to plot a segment of a conic curve”. Based on that manual construction one can demonstrate that (see Mortenson (1985)[74] for details):

Definition 3.3.1.1. *Given three points P, Q and T, there is a conic curve whose tangents at P and Q lie along the vectors $\overrightarrow{PT}$ and $\overrightarrow{QT}$ respectively. The conic is also tangent to a line parallel to $\overline{PQ}$, and offset a distance ρH perpendicularly to $\overline{PQ}$, with $\rho \in [0, 1]$. H is the perpendicular distance from $\overline{PQ}$ to T . The conic is*

Hyperbola, if $\rho \in \,]\,0.5,\ 1.0\,]$.

Parabola, if $\rho = 0.5$.

Ellipse, if $\rho \in [\,0,\ 0.5\,[$.



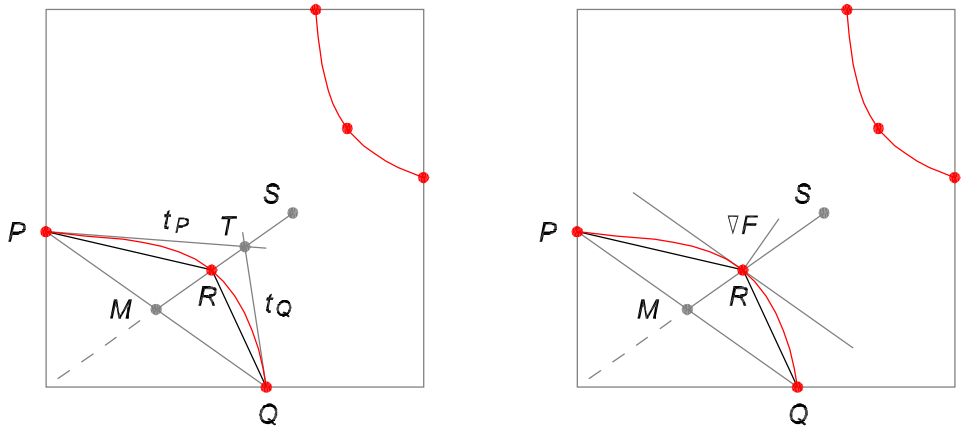
We want to approximate each hyperbolic arc as two straight line segments. Let us take as example the arc from P to Q of Figure 3.5(a). The point R is the shoulder point of the conic arc. It is the cornerstone of the method. It intersects the line $\overline{M\,T}$, where M is the mid-point of $\overline{P\,Q}$ and T is the intersection of the tangents at P and Q, respectively t_P and t_Q. Based on Definition 3.3.1.1, the tangent to the hyperbola at R is parallel to $\overline{P\,Q}$. Because the hyperbolic arc is convex, the shoulder point R is at maximum distance from $\overline{P\,Q}$ as Figure 3.5(b) also shows. This maximizes the area of the triangle delimited by P, R and Q, and therefore minimizes the area between the polyline $\overline{P\,R\,Q}$ and the contour function. So R is an optimal point to use as extra point. As a result, the hyperbolic arc is approximated by the polyline $\overline{P\,R\,Q}$ instead of $\overline{P\,Q}$.

The shoulder point is rather easy to compute. In the following we show two methods to compute it: as lying on $\overline{M\,T}$ as in Figure 3.5(a); or as lying on the line parallel to $\overline{P\,Q}$ as shown in Figure 3.5(b).

But first let us show that the product of the two first derivatives along the hyperbola is constant.

Proposition 3.3.1.2 (Constant product of first derivatives). *Given the hyperbola described by* F *of* (3.2)*, the following condition holds:*

$$F_x(x,y)\,F_y(x,y) = a\,b - c\,d\,, \quad \forall (x,y) \quad \textit{lying on the hyperbola.} \tag{3.5}$$



(a) (b)

Figure 3.5: Shoulder point R as extra point on the hyperbolic arc. It can be seen from two different perspectives. On (a) R is shown as lying on the line $\overline{M\,T}$ whereas on (b) as lying on the line parallel to $\overline{P\,Q}$.

Proof. Using the expressions of the first derivatives of F we obtain

$$\begin{aligned} F_x\,F_y &= (a + c\,y)\,(b + c\,x) \\ &= a\,b + a\,c\,x + b\,c\,y + c^2\,x\,y \\ &= a\,b + c\,(a\,x + b\,y + c\,x\,y) \\ &= a\,b - c\,d \qquad \text{by (3.2)} \end{aligned}$$

□

3.3.1.1 Shoulder point on $\overline{M\,T}$

Let us consider Figure 3.5(a). As shown in the following Proposition 3.3.1.3, $\overline{M\,S}$ contains $\overline{M\,T}$. Unlike T, the saddle point S is rather easy to compute. So we are looking for the intersection of $\overline{M\,S}$ with the hyperbola. Writing $\overline{M\,S}$ parametrically

$$x = x_M + t\,\alpha\,, \qquad y = y_M + t\,\beta\,, \tag{3.6}$$

with

$$\alpha = x_S - x_M\,, \qquad \beta = y_S - y_M\,, \qquad t \in [\,0,\,1\,]\,.$$

and using (3.2) yields

$$\begin{aligned} a\,(x_M + t\,\alpha) + b\,(y_M + t\,\beta) + c\,(x_M + t\alpha)(y_M + t\beta) + d = \\ (a\,x_M + b\,y_M + c x_M y_M + d) + t\,(a\alpha + b\beta + c\,(x_M\beta + y_M\alpha)) + t^2(c\alpha\beta) = \\ F(x_M, y_M) + t(a\alpha + b\beta + c(x_M\beta + y_M\alpha)) + t^2(c\alpha\beta) = 0\,. \end{aligned} \tag{3.7}$$

Therefore the computation of R involves solving the quadratic (3.7). Recall that the solution of interest is such that $t \in \,]0,\,1[$. Then we use the value of t in (3.6) to obtain R.

Proposition 3.3.1.3 (Points M, T and S are collinear). *Let $F\,(x, y) = a\,x + b\,y + c\,x\,y + d$ be the contour interpolant within the cell with all function values at cell vertices different from the isovalue $\Theta = 0$ (non-intersection vertices). Also, let*

- *P and Q be two points where $F\,(x, y) = 0$ cuts two cell edges, and lying on the same hyperbolic arc.*
- *M be the midpoint of $\overline{P\,Q}$.*
- *T be the intersection point of the two tangent lines to $F\,(x, y) = 0$ at P and Q, respectively t_P and t_Q.*
- *S be the saddle point, where both first derivatives F_x and F_y vanish.*

Then points M, T and S are collinear.

Proof. Since $\overline{M\,T}$ and $\overline{M\,S}$ have a common point (M), we simply need to show they have the same slope.

The implicit equations of tangent lines t_P and t_Q are

$$F_x^P\,(x - x_P) + F_y^P\,(y - y_P) = 0\,, \tag{3.8}$$

$$F_x^Q\,(x - x_Q) + F_y^Q\,(y - y_Q) = 0\,. \tag{3.9}$$

Rewriting (3.8) and (3.9) yields

$$F_x^P\,x + F_y^P\,y + \alpha = 0\,, \tag{3.10}$$

$$F_x^Q\,x + F_y^Q\,y + \beta = 0\,, \tag{3.11}$$

with

$$\alpha = -F_x^P\,x_P - F_y^P\,y_P\,, \tag{3.12}$$

$$\beta = -F_x^Q\,x_Q - F_y^Q\,y_Q\,. \tag{3.13}$$

Hence, from (3.10) and (3.11), the coordinates of their intersection point T are

$$x_T = \frac{F_y^P\,\beta - F_y^Q\,\alpha}{F_x^P\,F_y^Q - F_y^P\,F_x^Q}\,, \qquad\qquad y_T = \frac{F_x^Q\,\alpha - F_x^P\,\beta}{F_x^P\,F_y^Q - F_y^P\,F_x^Q}\,. \tag{3.14}$$

($F_x^P\,F_y^Q = F_y^P\,F_x^Q$ implies lines are parallel or possibly identical). Since the derivatives F_x and F_y at P and Q are

$$F_x^P = a + c\,y_P\,, \qquad\qquad F_y^P = b + c\,x_P\,, \tag{3.15}$$

and

$$F_x^Q = a + c\,y_Q\,, \qquad\qquad F_y^Q = b + c\,x_Q\,, \tag{3.16}$$

we can define the coordinates of M as

$$\begin{aligned} x_M &= \frac{x_P + x_Q}{2} & y_M &= \frac{y_P + y_Q}{2} \\ &= \frac{1}{2\,c}(F_y^P + F_y^Q - 2\,b)\,, & &= \frac{1}{2\,c}(F_x^P + F_x^Q - 2\,a)\,. \end{aligned} \tag{3.17}$$

Now, using (3.14), and (3.17) we deduce the slope of $\overline{M\,T}$

$$\begin{aligned} \frac{y_T - y_M}{x_T - x_M} &= \frac{\dfrac{F_x^Q\,\alpha - F_x^P\,\beta}{F_x^P\,F_y^Q - F_y^P\,F_x^Q} - \dfrac{1}{2\,c}\Big(F_x^P + F_x^Q - 2\,a\Big)}{\dfrac{F_y^P\,\beta - F_y^Q\,\alpha}{F_x^P\,F_y^Q - F_y^P\,F_x^Q} - \dfrac{1}{2\,c}\Big(F_y^P + F_y^Q - 2\,b\Big)} \\ &= \frac{2\,c\,(F_x^Q\,\alpha - F_x^P\,\beta) - (F_x^P\,F_y^Q - F_y^P\,F_x^Q)(F_x^P + F_x^Q - 2\,a)}{2\,c\,(F_y^P\,\beta - F_y^Q\,\alpha) - (F_x^P\,F_y^Q - F_y^P\,F_x^Q)(F_y^P + F_y^Q - 2\,b)}\,. \end{aligned} \tag{3.18}$$

Rewriting (3.12), (3.13) using (3.15), (3.16), and replacing in (3.18) above, yields

$$\frac{y_T - y_M}{x_T - x_M} = \frac{-\,3\,F_x^P\,F_y^P\,F_x^Q + 3\,F_x^P\,F_x^Q\,F_y^Q - {F_x^P}^2\,F_y^Q + F_y^P\,{F_x^Q}^2}{-3\,F_y^P\,F_x^Q\,F_y^Q + 3\,F_x^P\,F_y^P\,F_y^Q - F_x^P{F_y^Q}^2 + {F_y^P}^2F_x^Q}\,. \tag{3.19}$$

Next, the slope of $\overline{M\,S}$. The saddle point S has coordinates $(-b/c, -a/c)$. Using also (3.17), the slope of $\overline{M\,S}$ is

$$\begin{aligned} \frac{y_S - y_M}{x_S - x_M} &= \frac{-\dfrac{a}{c} - \dfrac{1}{2\,c}\Big(F_x^P + F_x^Q - 2\,a\Big)}{-\dfrac{b}{c} - \dfrac{1}{2\,c}\Big(F_y^P + F_y^Q - 2\,b\Big)} \\ &= \frac{F_x^P + F_x^Q}{F_y^P + F_y^Q}\,. \end{aligned} \tag{3.20}$$

Since we want both slopes to be equal then from (3.19) and (3.20) we deduce the following equation that has to be satisfied:

$$F_x^P\,F_x^Q\,F_y^Q\,F_y^Q - F_x^P\,F_y^P\,F_y^P\,F_x^Q + F_y^P\,F_x^Q\,F_x^Q\,F_y^Q - F_x^P\,F_x^P\,F_y^P\,F_y^Q = 0\,. \tag{3.21}$$

Using Proposition 3.3.1.2, so $F_x^P\, F_y^P = F_x^Q\, F_y^Q = a\,b - c\,d$, one can verify that (3.21) holds. □

3.3.1.2 Shoulder point on the line parallel to $\overline{P\,Q}$

Let us consider Figure 3.5(b). Based on Definition 3.3.1.1 the shoulder point has to lie on a line parallel to $\overline{P\,Q}$. At any point on the curve of $F = 0$, the gradient ∇F is always perpendicular to the curve. Therefore the dot product $\nabla F \cdot \overrightarrow{P\,Q}$ is zero. Using also Proposition 3.3.1.2, computing R requires to find $(x_R\,,\,y_R)$ such that:

$$F_x\,\alpha + F_y\,\beta = 0\,, \qquad \text{with } \overrightarrow{P\,Q} = \langle\,\alpha, \beta\,\rangle\ , \tag{3.22}$$

$$F_x F_y = a\,b - c\,d\,, \qquad \text{by Proposition 3.3.1.2}\,. \tag{3.23}$$

The solution is

$$F_x = \sqrt{-(a\,b - c\,d)\,\frac{\beta}{\alpha}}\,, \qquad F_y = \frac{a\,b - c\,d}{F_x}\,. \tag{3.24}$$

Once F_x and F_y are obtained, R is immediately found from (3.3) as:

$$x_R = \frac{F_y - b}{c}\,, \qquad y_R = \frac{F_x - a}{c}\,. \tag{3.25}$$

There are two solutions for F_x in (3.24), one positive and one negative. So we must choose the proper one. One way to do so is to make sure that R will lie between the midpoint of $\overline{P\,Q}$ and the saddle point S. Notice that the saddle point S is halfway between the two solutions we would have obtained for R. Indeed, (3.25) can be rewritten as

$$x_R = \frac{F_y}{c} + x_S\ , \qquad y_R = \frac{F_x}{c} + y_S\ . \tag{3.26}$$

3.4 Tests

We now present some contouring tests. The goal is to compare the three contour approaches discussed before: the basic method, the four triangles technique and our own approach based on shoulder points. The first and the last provide the same topology since they both use the asymptotic decider criterion. The second might lead to different topological solutions. In all the accuracy of the solutions will differ from method to method.

Assuming data linearity between adjacent cell points, we have formulated two tests of topographic elevation data. They are:

Antarctica elevation data.

This rectangular 2D data set depicts the elevation data of Antarctica. It is a data example included in IRIS Explorer, of resolution $[\,30 \times 36\,]$. We will show five contour levels: $\Theta = \{\,0.1,\ 17.6,\ 25.9,\ 31.9,\ 38.8\,\}$. Also, we will compare the results from the different methods.

Area of United States elevation data.

This rectangular 2D data set of resolution $[\,512 \times 512\,]$ corresponds to elevation data of an area of the United States of America, in a scale $1 : 250,000$. The data is available from the United States Geological Survey. We will show eight contour levels, from $\Theta = 500.25$ to $\Theta = 4000.25$, and equally spaced.

Let us start with the Antarctica map.

Figure 3.6 shows the five contour levels $\Theta = \{\,0.1,\ 17.6,\ 25.9,\ 31.9,\ 38.8\,\}$. Next, Figure 3.7 illustrates the differences between our new method and the basic one. The differences are represented by filled triangles delimiting both solutions. Notice that the end-points of the new polyline solution are simultaneously the end-points of the related straight line of the basic method. The topology is always the same. Next, Figure 3.8 superimposes both the new method and the four triangles technique. The cyan lines are from the four triangle technique whereas the remaining ones are from our method. If we look closely at the southeast region of the image, for contour level $\Theta = 17.6$, we find

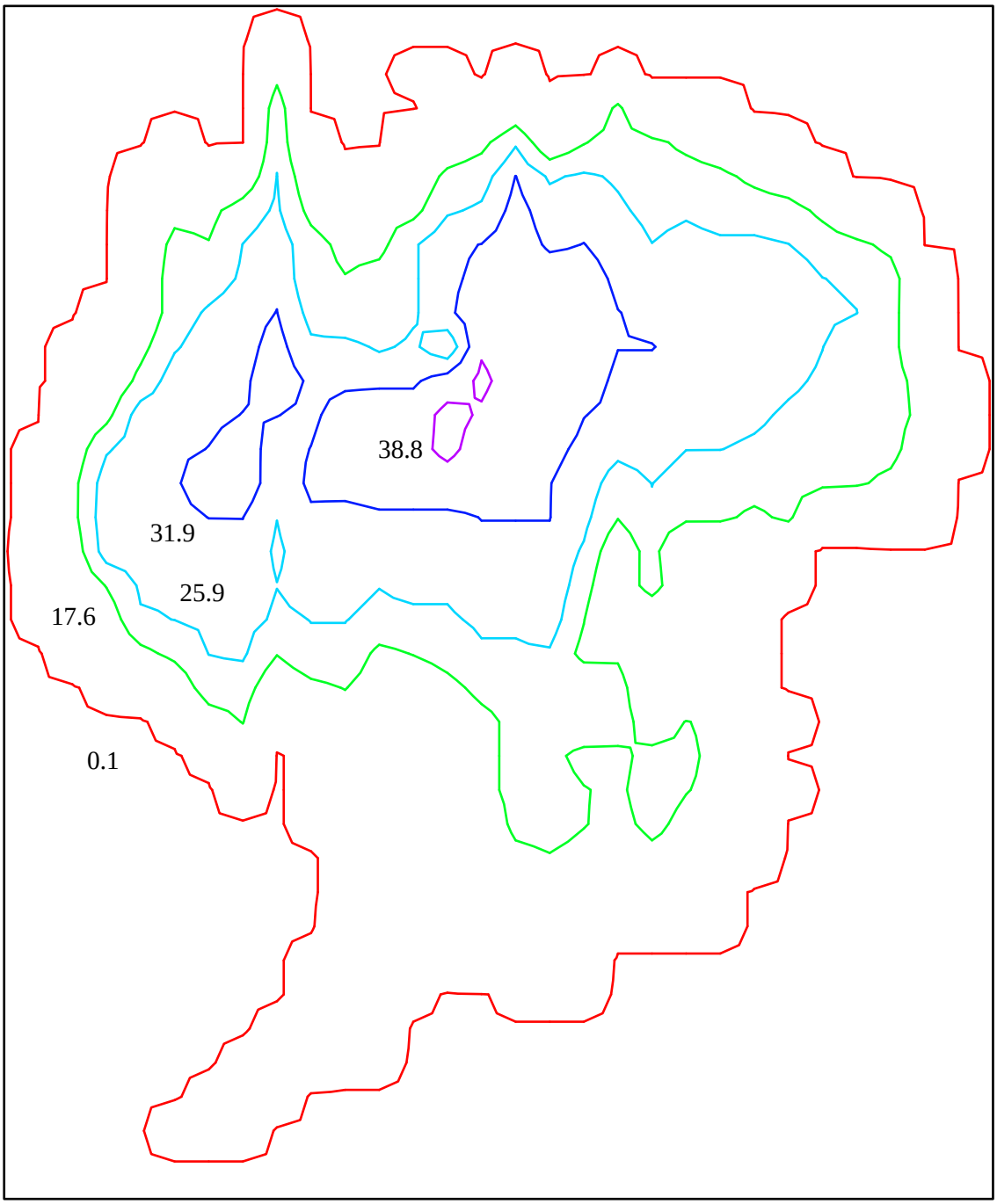


Figure 3.6: Antarctica map. Contour plot of five levels, using the new method. Data is of resolution $[30 \times 36]$.

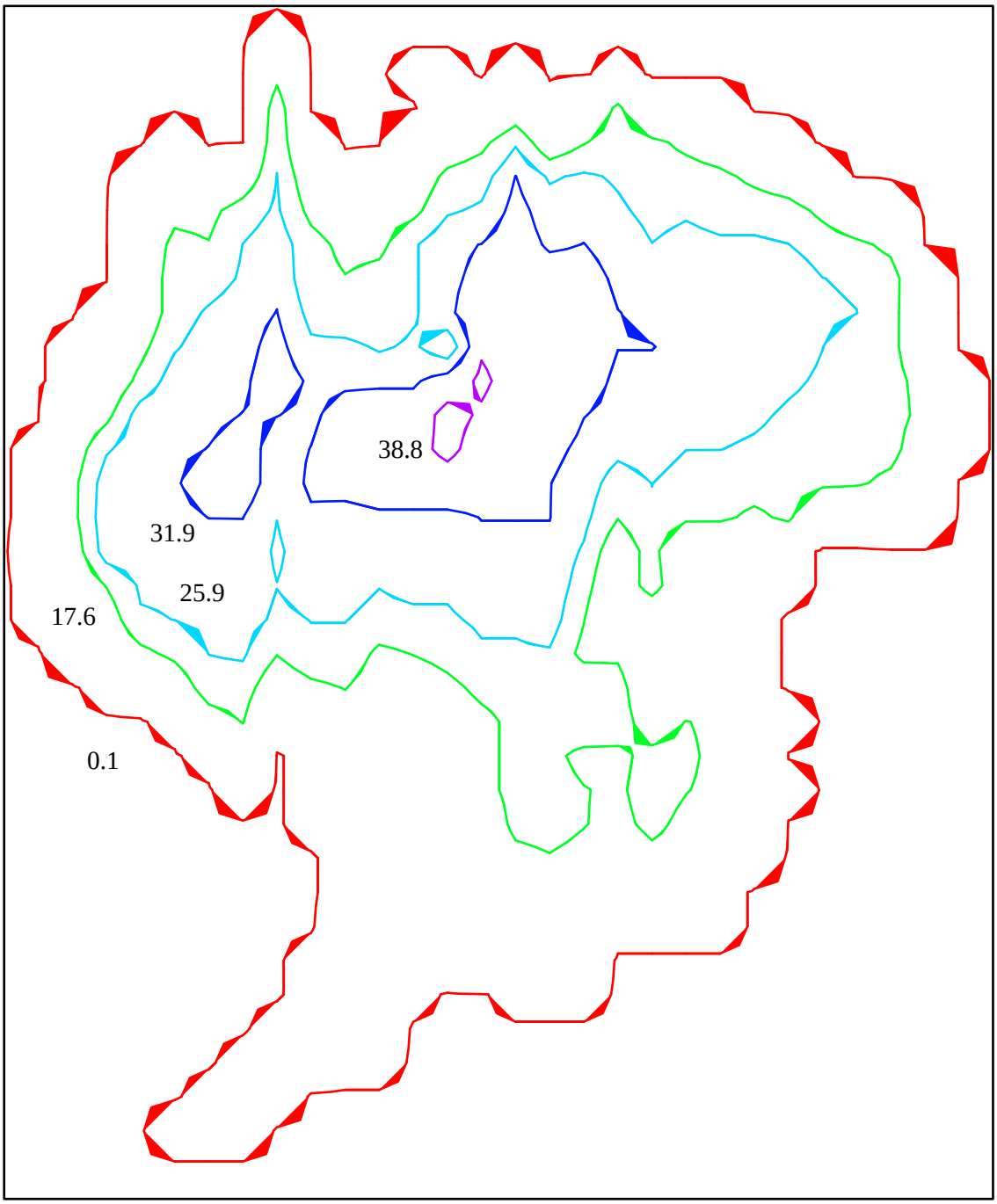


Figure 3.7: Antarctica map. Differences between the new method and the basic one are denoted by filled triangles delimiting both solutions. Data is of resolution $[30 \times 36]$.

different topologies in one cell. Figure 3.9 highlights the situation. We have identified the problem cell as having the following function values at the vertices: $f_{00} = 15.0$, $f_{10} = 21.0$, $f_{11} = 10.0$ and $f_{01} = 24.0$. The bilinear interpolant is

$$F(x\,y) = 6.0\,x + 9.0\,y - 20.0\,x\,y + 15.0 = 17.6 \tag{3.27}$$

The function value at the centre of the cell is $17.5 - 17.6 = -0.1$, whereas at the saddle point the value is $17.7 - 17.6 = +0.1$. These values have opposite sign so the methods draw the topologies differently. In the case of the four triangles method there are two points of the contour that are wrongly assumed to lie on the diagonal from $(0,\,1)$ to $(1,\,0)$.

3.5 Accuracy of Contouring

As underlined in Chapter Two, we are keen to give users some idea of the error committed while using a visualization technique. Although we claim that our new contouring method is more accurate than other methods — on the assumption that data is bilinear and free of errors — there will be always uncertainty since the hyperbolic contour is approximated by straight lines. To show the error committed we first establish an error metric and then map it to a visual paradigm.

3.5.1 Error Metrics

The metric used should reflect the kind of error we commit. Contouring is basically a 2D problem. A measure of the error is the deviation of the drawn contour from the exact contour line. This deviation is an *area* measurement. Figure 3.11(a) depicts the idea: the error is measured by the area between the hyperbola and its straight line approximation. Other metrics could have been used instead. For instance the difference of lengths of the hyperbola and its approximation. We believe however that area is by far the most appropriate metric for 2D contouring.

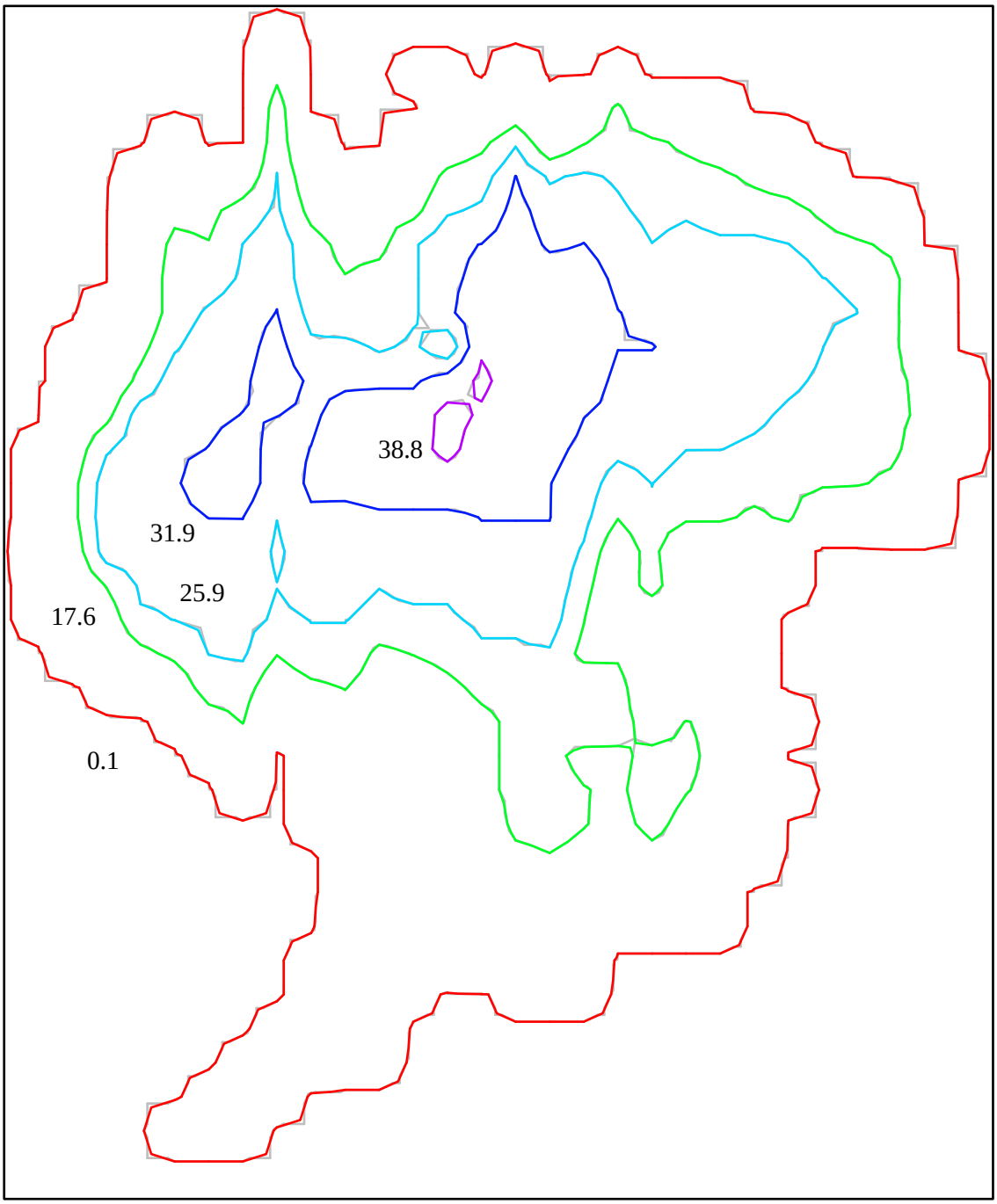


Figure 3.8: Antarctica map. Differences between the new method and the four triangles technique (grey lines). Data is of resolution $[30 \times 36]$.

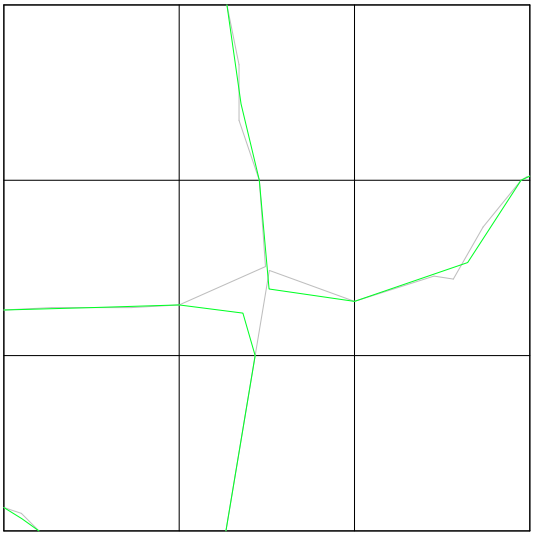


Figure 3.9: Topology from the new method (green) and the four triangles technique (grey). The four triangles technique gives a wrong topology for the cell in the middle.

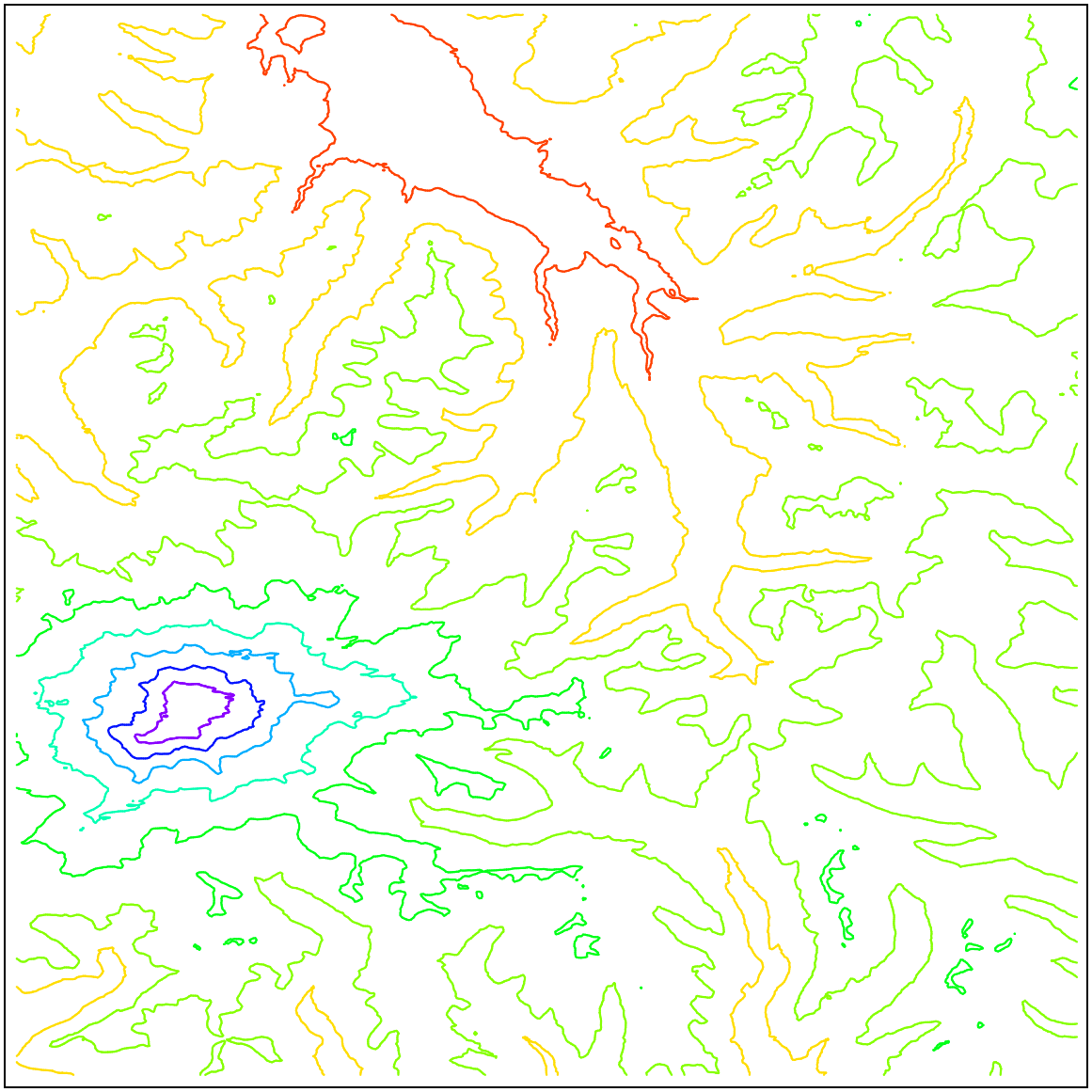
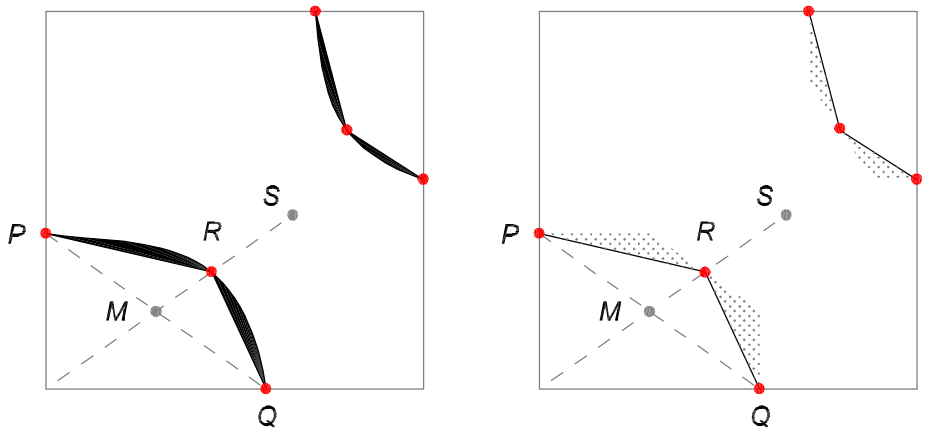


Figure 3.10: Digital elevation data in an area of United States. There are represented eight contour levels: from $\Theta = 500.25$ to $\Theta = 4000.25$, and equally spaced. Data is of resolution $[\,512 \times 512\,]$.



(a) (b)

Figure 3.11: Accuracy of the new method. On (a) one can see the error committed and on (b) the visual paradigm to represent it as random points within a well defined region.

To compute this metric one can rewrite the interpolant F of (3.2) as (with threshold value $\Theta = 0$):

$$y = -\frac{a\,x + d}{c\,x + b}, \tag{3.28}$$

and then use the integral of (3.28) to calculate the area ”under” the hyperbola. It yields

$$\int y\,dx = \frac{1}{c}\left[-a\,x + \left(\frac{a\,b}{c} - d\right)\,\ln\left|\,b + c\,x\,\right|\right]. \tag{3.29}$$

Hence the area between the hyperbola and the polyline can be easily calculated.

3.5.2 Visual paradigms

A visual paradigm should be relative to its purpose. So how should we convey the error metric we have established?

Since a contour plot is drawn on a 2D space and so easily perceived we can overlay geometric information on the plot itself. These extra geometries are random points in the vicinity of the contour plot towards the exact solution, confined within regions as depicted

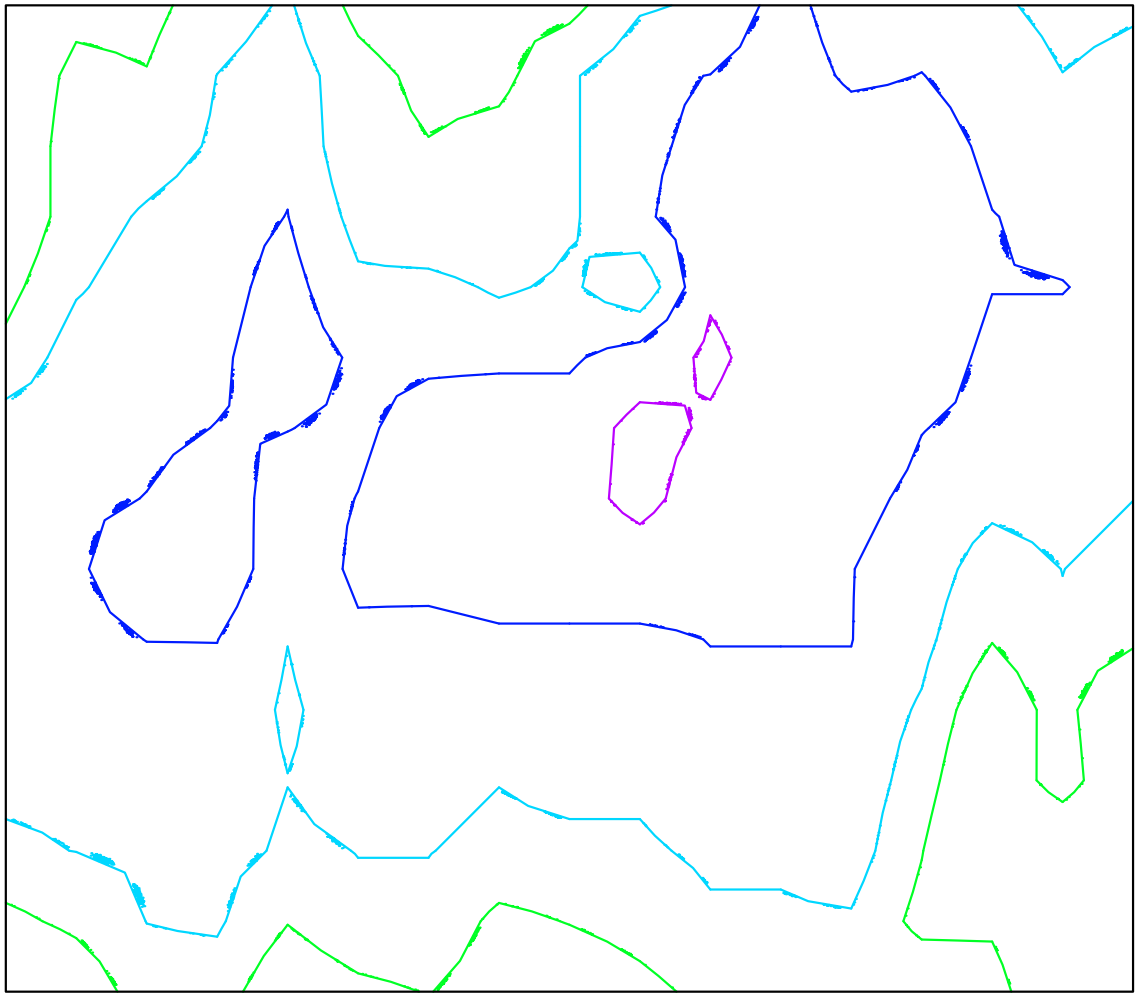


Figure 3.12: Accuracy of the new method in the Antarctica map. Random points are plotted in the vicinity of the contour plot, and towards the exact solution.

in Figure 3.11(b). It looks like there is a ”dust cloud” in the vicinity of the plot. Indeed the metaphor associated with random points is a scope of alternatives. The bigger the error the greater the area of random points. We provide a plot scale associating the maximum metric value to a maximum number of random plots. This controls the density of points and to some extent the cluttering of the final image. Figure 3.12 depicts the accuracy of the new method for the Antarctica map.

Alternatively one can think of mapping the metric to a solid but transparent background. For example colouring the related background area according to the metric at each cell. This is basically the colour bands technique, mentioned in Section 2.4.2 on page 17. The grey scale is the most appropriate since shades of grey often show vary-

ing quantities better than colour. But one has to make sure it does not conflict with the colouring of the contour plot. In both paradigms presented above there are no changes of the attributes of the contour plot such as the colour or the linetype.

3.6 Summary

We have presented a new method to draw contour lines on quadrangular grids, with a bilinear function as reference for each cell. Each hyperbolic arc is approximated by two straight line segments. The cornerstone of the method lies upon the common point joining the two segments — the shoulder point. We proved mathematically that this optimal point lies on the line joining the midpoint of the chord between the end points of the hyperbolic arc, and the saddle point of the bilinear function.

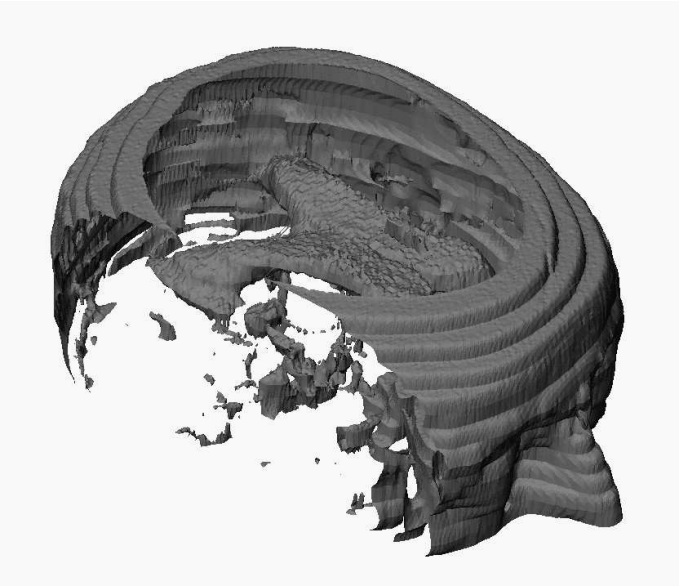
In order to provide correct topology, we use the asymptotic decider criterion. Positional continuity is guaranteed as well. In the end, at no substantial computational cost we obtain a more accurate solution in comparison to two classic methods: the basic method and the four triangles method. None of them is a satisfactory contouring method. The basic method just uses one line segment to approximate the hyperbolic arc; the four triangles method uses two line segments to approximate the hyperbolic arc but neither all the end points lie on the hyperbolic arc, nor it does always provide correct topology.

Another aspect under study was the representation of accuracy. There is always some uncertainty, regardless the quality of the algorithms. We have defined an exact error metric: the area whose boundary is defined by the contour solution and the representation of its approximation. Then, for each grid cell we mapped this value using a metaphor of ”dust cloud”: random points are plotted in the vicinity of the approximation drawn, and towards the exact contour.

Chapter

FOUR

Isosurfacing



Isosurfacing, sometimes called surface fitting, extracts boundary surfaces from a volumetric scalar data. This is a major facet in scientific visualization. The boundary surface separates points with values greater or equal to a threshold from those with values less than. The result is the so-called isosurface, depicting where data values of a constant threshold lie.

In this chapter we will discuss in depth the most classic isosurfacing method: the

Marching-Cubes. First, we present a complete study of the method. Some aspects are clarified and corrected, with emphasis on the concepts of topological polygon and topological correctness. Then we propose a more accurate Marching-Cubes. This new proposal relies mostly on three types of points that we define: face shoulder points, inflection points, and bi-shoulder points. Next, we suggest some ways to show accuracy in isosurfacing. Finally we discuss the overall results and point out research directions.

4.1 Marching-Cubes

In order to show a particular isosurface from a regularly sampled data set over a domain D we construct a piecewise approximation to the isosurface. Formally the problem is to represent

$$F_\Theta = \{\, (x,\, y,\, z) \mid F(x,\, y,\, z) = \Theta\,, \quad (x,\, y,\, z\,) \in D\,\}\,. \tag{4.1}$$

The Marching-Cubes method (hereafter called the MC method) was introduced by Lorensen and Cline[67] in 1987. It uses the cube as computational cell (eight data values corresponding to the eight vertices of the cube) and they are processed one by one. Then all the individual contributions are assembled together to obtain the whole isosurface.

The data is assumed to vary linearly along the cell edges. If at one endpoint of an edge the value is greater than the isovalue Θ (let us call it a positive vertex) and smaller at the other endpoint (a negative vertex), then there is obviously a point on the edge such as the value is Θ. It is a so-called *intersection point*. This point will then be used as vertex of a polygonal approximation to the isosurface. A polygon delimited by a set of intersection points is called a *topological polygon*. Note that these polygons are in general non-planar.

There are up to $2^8 = 256$ possible patterns for the intersection between the isosurface and the cubic cell. But by taking into account complementarity and also rotational and reflection symmetries in the cube, this number can be reduced to fourteen. Figure 4.1 shows these basic patterns, or *configurations* as we name them. These configurations give rise to the topological polygons.

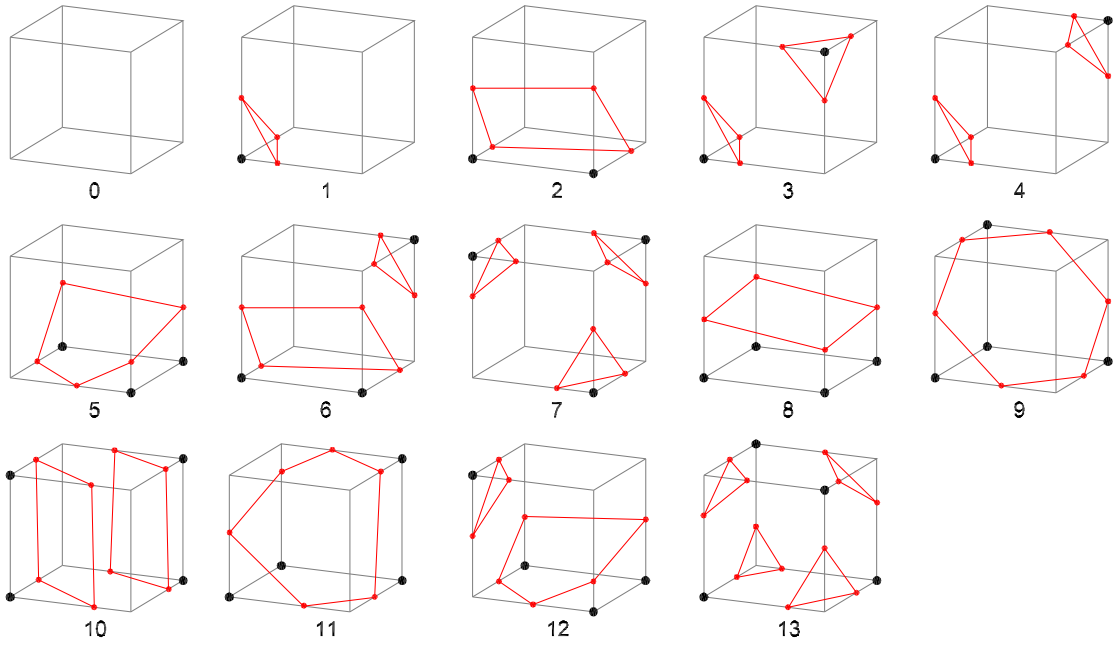


Figure 4.1: Basic configurations in the Marching-Cubes method, and examples of topological polygons that would result.

In conclusion, the intersection of the isosurface and the cube cell will partition the cell vertices in such a manner that positive vertices are separated from the negative ones. Even more, it may partition the positive vertices themselves into several disjoint subsets. The same holds for the negative vertices.

4.1.1 Ambiguities

Despite its simplicity and attractiveness a certain number of anomalies soon were to be highlighted. The first problem, mentioned by Dürst,[30] was a potential flaw as a consequence of mismatched patterns between adjacent cells in the data set. Figure 4.2 illustrates such a problem: discontinuity between cells arises and so flaws in the surface occur. The problem is that there are cases where it is unclear how to define the topology. The polygonal approximation to the isosurface must be continuous (let us just consider *positional continuity*, C^0) and topologically correct. As van Gelder and Wilhelms pointed out,[111] it is C^0 continuous between cells if and only if each edge is shared by exactly two polygons, except for edges that lie on the boundary of the volume, which must occur

exactly in one polygon. The topology is certainly incorrect if not continuous. Yet being C^0 continuous does not necessarily mean it is topologically correct. It turned out that further topological analysis was required. More topological polygons were needed in addition to those indicated in Figure 4.1.

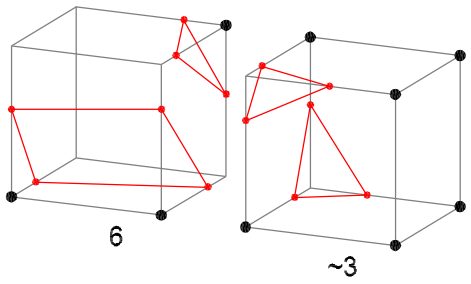


Figure 4.2: Flaws in the MC method. They might arise if ambiguities are not dealt with properly. This example shows configuration 6 and the inverse of configuration 3 as adjacents.

One way to attempt to resolve ambiguity is to resample the data until the ambiguity disappears. This may work, in particular if the underlying function is known. On the other hand, resampling may be impossible to carry out, for instance if data is from physical measurements which cannot be repeated.[111] Therefore different approaches are welcome.

Ambiguities can arise both in the faces and in the interior of the cell. Let us start by discussing ambiguity in the cell faces.

In the previous Chapter, Section 3.2.1 on page 42, we have discussed the issue of ambiguity in 2D. To recall, disambiguation in 2D is a matter of choice of the pairwise connection in faces with four intersection points, the so-called *ambiguous faces*. We mentioned two methods to resolve facial ambiguity: the *four triangles technique* (Section 3.2.1 on page 43), used in the context of isosurfacing by Wyvill *et al.* ,[126] and the *asymptotic decider*[78] (Section 3.2.1 on page 43). Both methods ensure C^0 continuity between cells. But the four triangle method does not always guarantee facial topological correctness. Fortunately this is achieved by the asymptotic decider.

The asymptotic decider extends the configurations of Figure 4.1 which have one or more ambiguous faces: 3, 6, 7, 10, 12 and 13. The following Figures 4.3 and 4.4 show

the results as presented by Nielson and Hamman.[78] In general for each of those problem configurations there are up to 2^n possibilities to consider, n being the number of ambiguous faces. But again this number can be reduced if complementarity and rotational and reflection symmetries are considered. For example, in configuration 7 the number of cases is $2^3 = 8$ but these reduce to the four cases shown in Figure 4.3.

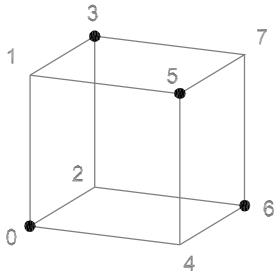
In respect to this work of Nielson and Hamman we would like to draw attention to three aspects. Firstly, a new subcase not reported by them should be mentioned: 13.j (see Figure 4.4), which can not be included in any other class of configuration 13. Figure 4.5 depicts an example of subcase 13.j. Secondly, we can rule out the practical existence of subcases 13.f, 13.g and 13.h, which Nielson and Hamman suggest are possible. Recall that the asymptotic decider uses the function value of the saddle point in a face to establish the pairwise connection. If we assume that the subcase 13.f shown in Figure 4.4 exists then we conclude that the saddle point value on both top and bottom faces is positive whereas on both front and back faces the saddle point value is negative. This is contradicted by the following Proposition 4.1.1.1. The same argument is applied to subcases 13.g and 13.h.

Finally we note that if we follow the policy of equivalence under complementarity and rotation (as it has been used to obtain the MC configurations so far) then the grouping of subcases can go even further as far as configurations 10, 12 and 13 are concerned. These are configurations with equal number of positive and negative vertices. For instance, Nielson and Hamman distinguish 10.a and 10.c (see Figure 4.3) presumably because they illustrate different choices of the asymptotic decider. But if we take configuration 10.c and multiply the corner values by -1 (complementarity) and then rotate around the vertical axis to the left by ninety degrees we will obtain configuration 10.a. So we consider them as just one subcase. Moreover we shall emphasise later in this Chapter that these merging subcases are equivalent in terms of polygons generated, as indeed is obvious from Figures 4.3 and 4.4. In summary, the subcases that can be merged are: 10.a–10.c, 10.b–10.d, 12.a–12.c, 12.b–12.d, 13.b–13.c, and 13.d–13.e. We will refer to these pairs as 10.ac, 10.bd and so on.

Proposition 4.1.1.1. *Given the eight corner data values of a cube cell with MC configuration 13, and considering two pairs of opposite faces, it is impossible that for both faces in one pair the function value at the respective face saddle point is greater than zero and for both faces in the other pair that value is smaller than zero.*

Proof. Let us consider two pairs of opposite faces: $A = \{ \mathcal{F}_{A_1}, \mathcal{F}_{A_2} \}$ and $B = \{ \mathcal{F}_{B_1}, \mathcal{F}_{B_2} \}$. In accordance with the MC configuration 13 represented below (positive vertices are 0, 3, 5 and 6 whereas the remaining 1, 2, 4 and 7 are negative), the corner data values f_i of each face are:

$$\mathcal{F}_{A_1} = \{ f_0, f_1, f_4, f_5 \} ,$$
$$\mathcal{F}_{A_2} = \{ f_2, f_3, f_6, f_7 \} ,$$
$$\mathcal{F}_{B_1} = \{ f_0, f_2, f_4, f_6 \} ,$$
$$\mathcal{F}_{B_2} = \{ f_1, f_3, f_5, f_7 \} .$$



The function values at the face saddle points are (see previous Chapter, Section 3.2 on page 39, in particular (3.4)):

$$F_{A_1} = \frac{f_0 f_5 - f_1 f_4}{(f_0 + f_5) - (f_1 + f_4)} , \tag{4.2}$$
$$F_{A_2} = \frac{f_2 f_7 - f_3 f_6}{(f_2 + f_7) - (f_3 + f_6)} , \tag{4.3}$$
$$F_{B_1} = \frac{f_0 f_6 - f_2 f_4}{(f_0 + f_6) - (f_2 + f_4)} , \tag{4.4}$$
$$F_{B_2} = \frac{f_1 f_7 - f_3 f_5}{(f_1 + f_7) - (f_3 + f_5)} . \tag{4.5}$$

Let us assume by hypothesis that F_{A_1} and F_{A_2} are negative whereas F_{B_1} and F_{B_2} are positive. Also, taking into consideration the signs of the corner values f_i we conclude that the denominators on the right-sides of (4.2) and (4.4) are positive, whereas those of (4.3)

and (4.5) are negative. Therefore the following conditions should hold:

$$f_0\, f_5 < f_1\, f_4\ , \tag{4.6}$$

$$f_2\, f_7 > f_3\, f_6\ , \tag{4.7}$$

$$f_0\, f_6 > f_2\, f_4\ , \tag{4.8}$$

$$f_1\, f_7 < f_3\, f_5\ . \tag{4.9}$$

Using (4.8) and (4.9) and again taking into account the signs of f_i we can write that

$$f_0\, f_5 > f_1\, f_4\ \frac{f_2\, f_7}{f_3\, f_6}. \tag{4.10}$$

By (4.7) and since both numerator and denominator are positive the term $(f_2\, f_7)\,/\,(f_3\, f_6)$ in (4.10) is greater than one. In that case the conditions of (4.10) and (4.6) are contradictory. So the set of initial assumptions is incorrect. □

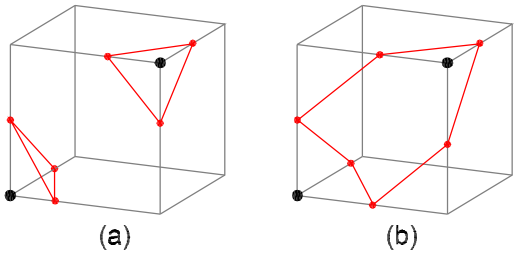
Now let us extend the problem of ambiguity to the interior of the cell. The asymptotic decider is only concerned with ambiguities in the cell faces. Furthermore, a cell can be ambiguous without having ambiguous faces. For example, configuration 4 has no ambiguous faces but may, or may not, lead to a *tunnel* as shown in Figure 4.6 (recall that data varies trilinearly within the cell). Nevertheless this particular configuration does not produce discontinuity between cells because there are no ambiguous faces.

In conclusion, ambiguous cells are from configurations which include ambiguous faces, *i.e.* 3, 6, 7, 10, 12, 13; and from configuration 4.

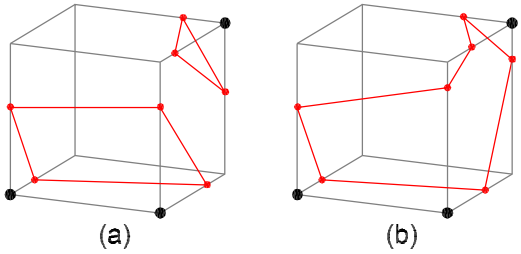
To detect the existence of internal tunnels, Natarajan[76] proposed a similar method to the asymptotic decider but assuming trilinearity within the cell. Basically he extended the concept of 2D saddle point to what he called a *body saddle point*: a point such that all the three first derivatives vanish. He indicated that internal tunnels can appear in configurations 4, 6, 7, 10, 12 and 13. We were not able to reproduce tunnels for configuration 13. We discuss the problem of tunnels in more detail in Section 4.2.2.3.

Table 4.1 summarises our study of the MC algorithm, and the topological correct-

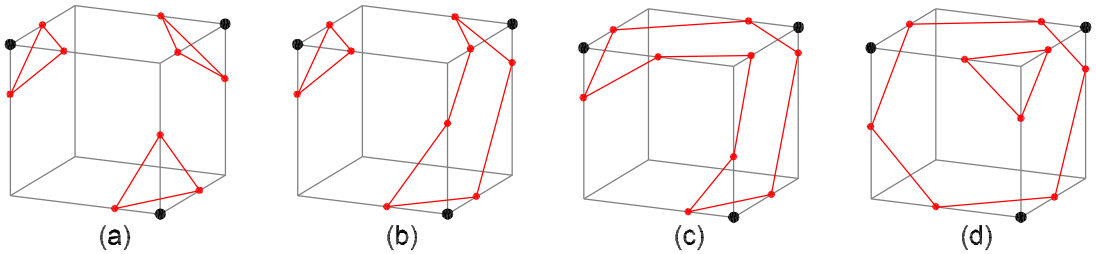
Configuration 3 :



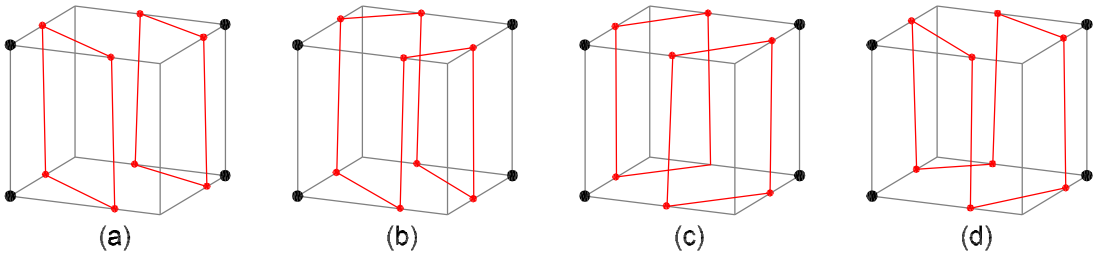
Configuration 6 :



Configuration 7 :



Configuration 10 :



Configuration 12 :

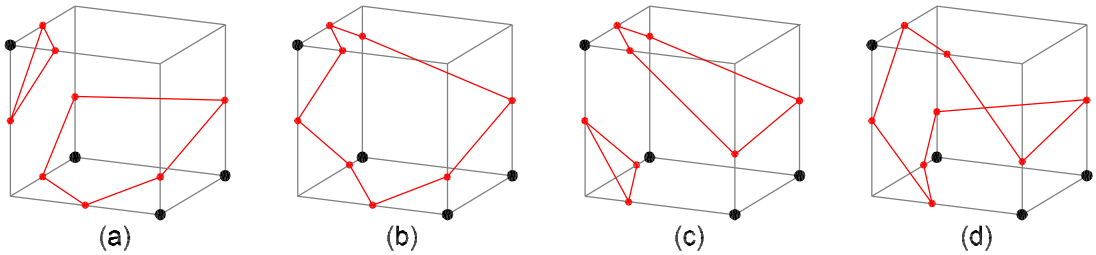


Figure 4.3: Asymptotic decider:[78] extending the MC configurations 3, 6, 7, 10 and 12. Notice that using again complementarity and rotational and reflection symmetries, one can consider some subcases as the same. For example, 10.c can be obtained by first complementing 10.a and then rotating around the vertical axis to the right by ninety degrees. Pairs to be grouped are 10.a-10.c, 10.b–10.d, 12.a–12.c and 12.b–12.d.

Configuration 13 :

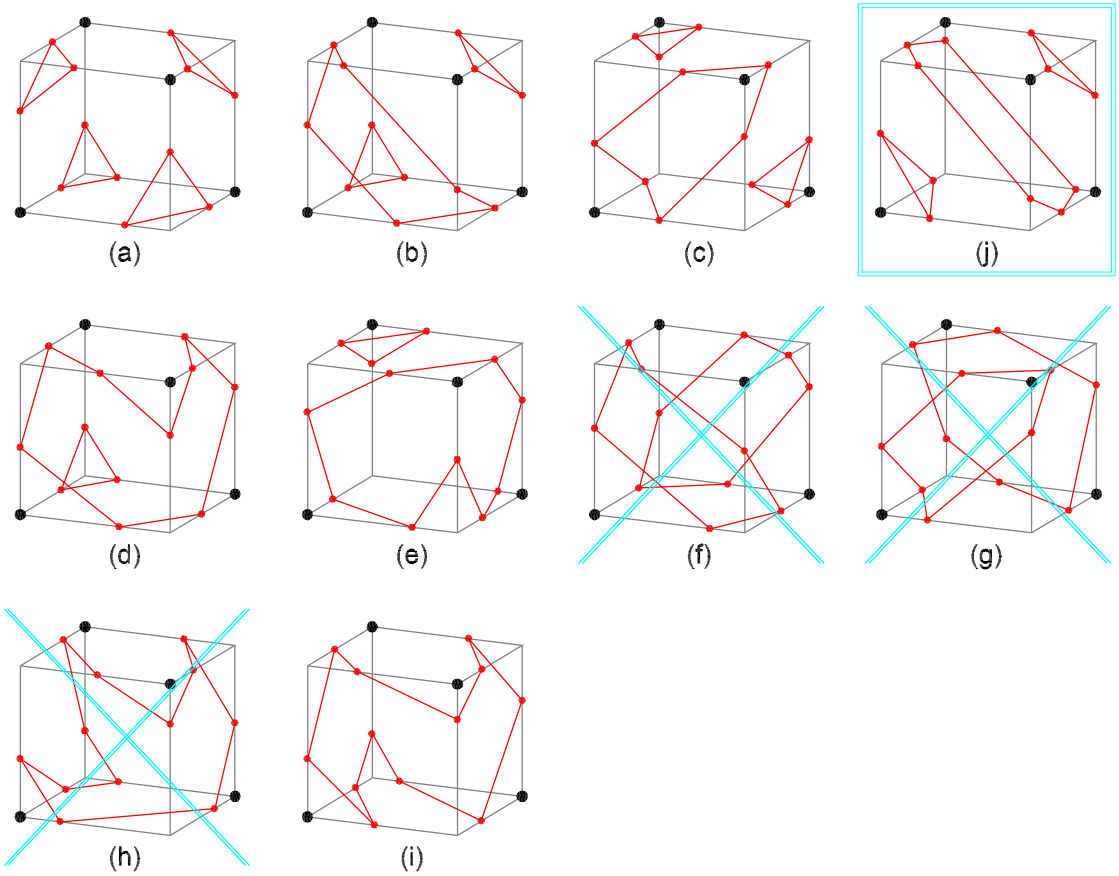


Figure 4.4: Asymptotic decider:[78] extending the MC configuration 13. Notice that we have just appended subcase 13.j and on the basis of Proposition 4.1.1.1 we rule out subcases 13.f, 13.g and 13.h. Also, taking into consideration complementarity and rotational and reflection symmetries as before, the following pairs of subcases can be considered as just one: 13.b-13.c and 13.d–13.e.

xyz	F_{xyz}
000	1.0
001	-3.0
010	-1.5
011	2.0
100	-3.0
101	2.0
110	2.0
111	-1.0

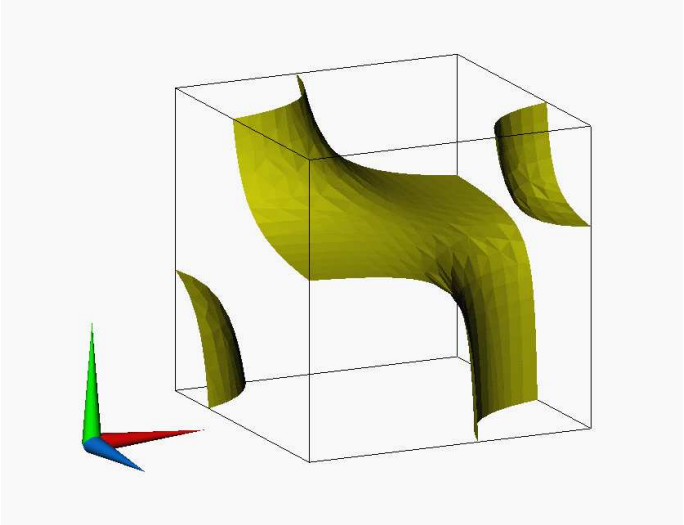
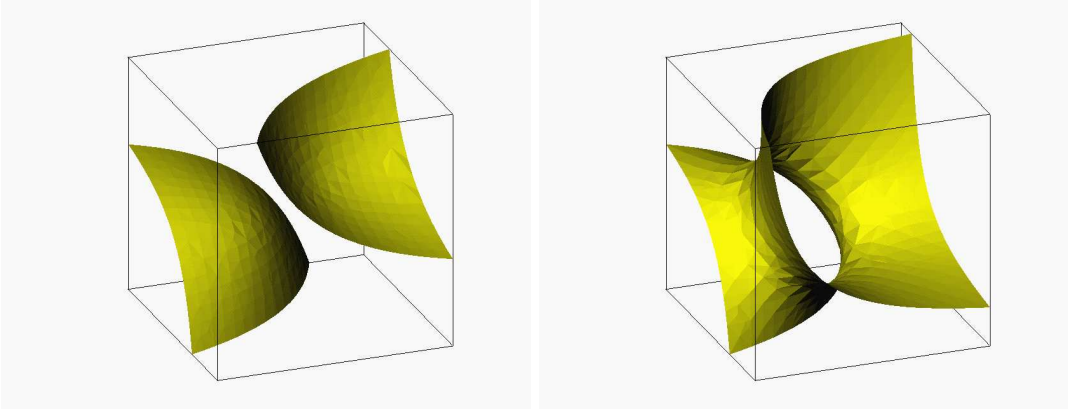


Figure 4.5: An example of subcase 13.j.



(a) (b)

Figure 4.6: Ambiguity in the interior of the cell. This is configuration 4 which has no ambiguous faces. On (a) we have the representation matching the classic MC configuration (see Figure 4.1). But on (b) the isosurface resembles a tunnel so this situation has to be considered as well.

ness of isosurfacing. It specifies the topological polygons which arise from application of the asymptotic decider (with our own corrections applied as just described). To help further discussions let us introduce at this point three concepts. They are: *loop-back faces*, *classification of topological polygons* and *equivalence of topological polygons*.

Definition 4.1.1.2 (Loop-back face). *An ambiguous face, i.e. with four intersection points, that is intersected (twice) by just one topological polygon is called a loop-back face (van Gelder and Wilhelms[111] have used the word* overworked *for the same concept but we believe our term is more expressive. See Figure 4.7).*

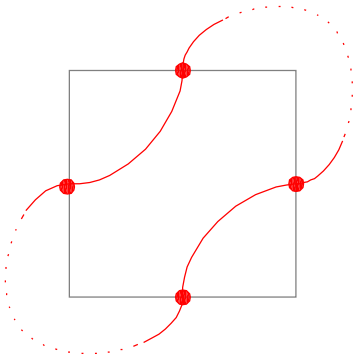


Figure 4.7: Loop-back face. The topological polygon intersects the face four times: in-out once and then again in-out.

Definition 4.1.1.3 (Topological polygons classification). *A topological polygon P is a polygon delimited by a set of intersection points. We denote them by the number of intersection points involved i, and the number of loop-back faces involved, j. But two loop-back faces that are opposite just count once. The notation for this classification is $P_{i/j}$.*

Definition 4.1.1.4 (Equivalent topological polygons). *Topological polygons with the same classification are said to be equivalent.*

Example 4.1.1.1. *Based on Figure 4.3,*

- *the front-face of configuration 3.b is a loop-back face;*

– *the notation for the topological polygon of configuration 3.b is $P_{6/1}$. For both 10.d and 12.d there are eight intersection points and two loop-back faces. But in 10.d the loop-back faces are opposed so the notation is $P_{8/1}$. Whereas they are not in 12.d, so the notation is $P_{8/2}$;*

– *all the topological polygons in configuration 3.a and 7.a are equivalent. They all have notation $P_{3/0}$.*

MC configuration	Ambiguity			Partitioning to polygons
	faces	cell	internal tunnel	
0				
1				one $P_{3/0}$
2				one $P_{4/0}$
3	one	yes		two $P_{3/0}$ or one $P_{6/1}$
4		yes	possible	two $P_{3/0}$
5				one $P_{5/0}$
6	one	yes	possible	$\{$one $P_{3/0}$, one $P_{4/0}\}$ or one $P_{7/1}$
7	three	yes	possible	three $P_{3/0}$ or $\{$one $P_{3/0}$, one $P_{6/1}\}$ or one $P_{9/3}$ or $\{$one $P_{3/0}$, one $P_{6/0}\}$
8				one $P_{4/0}$
9				one $P_{6/0}$
10	two	yes	possible	two $P_{4/0}$ or one $P_{8/1}$
11				one $P_{6/0}$
12	two	yes	possible	$\{$one $P_{3/0}$, one $P_{5/0}\}$ or one $P_{8/2}$
13	six	yes		four $P_{3/0}$ or $\{$two $P_{3/0}$, one $P_{6/1}\}$ or $\{$one $P_{3/0}$, one $P_{9/3}\}$ or one $P_{12/3}$ or $\{$two $P_{3/0}$, one $P_{6/0}\}$.

Table 4.1: Isosurfacing based on the MC method with application of the asymptotic decider. For any configuration of Figure 4.1, it is indicated the number of ambiguous faces, whether the cell is ambiguous or not and also if an internal tunnel is possible. On top of that, it is shown all the partitions of intersection points leading to topological polygons. To cite one example, there are two situations to consider on configuration 6 (see Figure 4.3): one polygon $P_{3/0}$ and one polygon $P_{4/0}$ or else just one polygon $P_{7/1}$. The notation used is of above Definition 4.1.1.3.

More techniques have been proposed to resolve ambiguity. Most of them require information beyond the extent of the cell being processed. For example, one proposal is to use a tricubic function within the cell that is C^1 continuous, based on values from the $[4 \times 4 \times 4]$ region of surrounding cells.[111] This is very expensive just to be used for disambiguity. Moreover, it has to be applied to all cells not just the ambiguous ones since discontinuities will arise if linear interpolation is applied in a non-ambiguous cell and cubic interpolation in an adjacent and ambiguous cell. Others use gradient information at the vertices of cells.[111] Indeed, the gradient highlights the behaviour of a function across the domain (for example, the gradient direction is normal to the surface and its magnitude indicates how rapidly the function changes).

One famous method emerged in the wake of solving ambiguity in MC: the Marching-Tetrahedra, which was mentioned in Section 2.4.5.2. Each cubic is decomposed into tetrahedra. Therefore within each tetrahedron the isosurface is correctly drawn as a plane, if a linear model is assumed. But we should point out that if the raw data set is a rectangular grid and the data is assumed to vary trilinearly within the cubic cell, as is the assumption in the MC, then its decomposition into tetrahedra may raise some concerns. First, it is incorrect to assume linear variation of data along the edges of the tetrahedra. Second, as a consequence no claim can be made that ambiguities of MC are automatically solved as result of decomposing into tetrahedra and then applying Marching-Tetrahedra.[81,130]

We find in the literature different proposals for the number and shape of the tetrahedrons.[15,42,44,85] The common numbers are five, six, twelve or even twenty-four. The orientation of the tetrahedra affects the final image as Figure 4.8 indicates. The aim is to achieve a regular and symmetric tesselation.

Yet due to the finer mesh resolution and consequently an increased number of triangles generated, the images from Marching-Tetrahedra are in general better than those provided by MC.

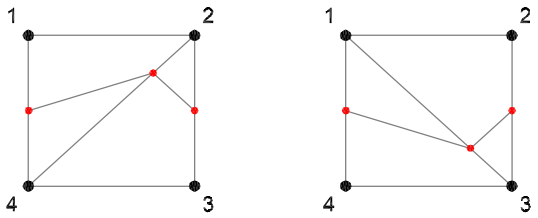


Figure 4.8: Decomposition into tetrahedra. Face of a cube, and threshold value of 2.5. The choice of diagonal orientation affects the final image, maybe leading to ”bumps”. To mantain consistency throughout the cells, while in 2D the choice can be arbitrary, in 3D it is constrained by the neighbouring cell.[94]

4.1.2 Surface rendering

Once the topological polygons are defined the next step is to provide geometric information to be mapped into standard graphics primitives. Normally triangulation is required. No matter the methodology used, the final surface has to be at least C^0 continuous and not self intersecting.

Many solutions have been proposed. A straightforward one is to define chords linking (non-adjacent) vertices in order to have adjacent non-intersected triangles that form a patch. But, very important, these chords are *not allowed to lie on faces*. If that was the case the face topology would be incorrect and so even C^0 continuity between cells is not achieved. Another method is to add an extra point internal to the cell — this can be the centroid of the polygon — and then define chords from that point to the vertices of the polygon.

Since no in-face chords are allowed, Nielson and Hamann[78] pointed out that some subcases depicted in Figures 4.3 and 4.4 were impossible to triangulate unless an extra point in the interior of the cell was added. These are subcases 7.c, 10.bd, 13.i and the polygon with nine vertices in 13.de. van Gelder and Wilhelms[111] also noticed that polygons in 13.f and 13.g require extra vertices to prevent self-intersection. But as we have shown before, 13.f and 13.g are impossible. They argued that as long as a cell contains two or more overworked (loop-back) faces then extra vertices are required. Centroids were then a suitable choice to be considered. However this is contradicted by triangulation of 12.bd delivered by Nielson and Hamann,[78] where no extra vertex was used. So

the correct condition for requiring an extra vertex is:

Proposition 4.1.2.1. *In a cell, triangulation of a topological polygon as established by the asymptotic decider requires an extra vertex if the following holds:*

- *it has edges from more than two overworked faces, as in 7.c, 13.de, 13.i;*
- *or it has edges from two but opposed overworked faces, as in 10.bd.*

In respect to those situations where tunnels might appear, triangulation is a rather difficult task and so it should be dealt with separately. As mentioned earlier, the method introduced by Natarajan[76] allows us to spot such situations. Yet no less important is how triangulation is carried out. To show its importance let us look at his example of configuration 6, depicted in Figure 4.9. There, the tunnel is supposed to be represented by one triangle plus three quadrilaterals. The hatched areas attempt to indicate the interior of the tunnel. But the triangulation of the quadrilateral that lies on the ambiguous face is impossible unless an extra point in the interior of the cell is added. This is due to the fact that no in-face triangles are allowed.

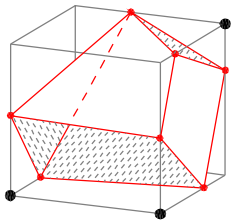


Figure 4.9: Tunnel in configuration 6 as referred by Natarajan.[76] How to triangulate the quadrilateral that lies on the ambiguous cell face, the right one? Since no in-face triangles are allowed then an extra point in the interior of the cell is required.

Finally, at this stage of triangulation it is important to use information that can enhance the visual appearance of the surface. That is the case of normals to the triangles used by shading algorithms. These normals are usually obtained by trilinear interpolation of the normals at the cell vertices. Of course this has to be available from the data set.

4.2 Accurate Marching-Cubes

From our work in Chapter Three on contouring, and from our understanding of the MC method in the previous Section, we are now able to develop a variant of the MC method which is a more accurate representation of the trilinear interpolant.

But first the formalisation of the problem. We aim to represent a piecewise trilinear interpolant F that fits scalar data values $F_{x,y,z}$ at the vertices of each cubic cell in a 3D rectilinear grid, as defined by (4.11). For the sake of clarity the cell is parametrised over the interval $[0,1]$ in $x-$, $y-$ and $z-$ directions and the threshold value for F is considered as 0. F can be written as:

$$\begin{aligned} F(x,y,z) \;=\; & F_{000}\;(1-x)\;(1-y)\;(1-z) \;+ \\ & F_{001}\;(1-x)\;(1-y)\;z \;+ \\ & F_{010}\;(1-x)\;y\;(1-z) \;+ \\ & F_{011}\;(1-x)\;y\;z \;+ \\ & F_{100}\;x\;(1-y)\;(1-z) \;+ \\ & F_{101}\;x\;(1-y)\;z \;+ \\ & F_{110}\;x\;y\;(1-z) \;+ \\ & F_{111}\;x\;y\;z \;. \end{aligned} \tag{4.11}$$

An alternative representation of (4.11) is

$$\begin{aligned} F(x,y,z) = \Delta^{000} + \Delta^{100}\,x + \Delta^{010}\,y + \Delta^{001}\,z \\ + \Delta^{110}\,x\,y + \Delta^{101}\,x\,z + \Delta^{011}\,y\,z + \Delta^{111}\,x\,y\,z \;, \end{aligned} \tag{4.12}$$

where $\Delta^{i,j,k}$ is the forward difference operator for triple indices, *i.e.* $\Delta^{000} = F_{000}$, $\Delta^{100} = F_{100} - F_{000}$, $\Delta^{110} = F_{110} - F_{100} - F_{010} + F_{000}$ and $\Delta^{111} = F_{111} - F_{110} - F_{101} - F_{011} + F_{100} - F_{010} - F_{001} - F_{000}$. To simplify the notation even more we will replace the forward difference operators by letters ($\Delta^{000} = a$, $\Delta^{001} = b$, $\Delta^{010} = c$ and so on). Therefore

$$F(x,y,z) = a + e\,x + c\,y + b\,z + g\,x\,y + f\,x\,z + d\,y\,z + h\,x\,y\,z \;. \tag{4.13}$$

Having the trilinear interpolant F of (4.13) as reference, the main objectives of our MC variant are:

1. To ensure positional continuity of the isosurface across adjacent cells and so correct topology in the cell faces.

2. To represent the trilinear interpolant in the interior of the cell as accurately as possible. In particular, the algorithm should generate an approximation to the actual isosurface which has the same number of pieces, and which has similar shape. As a consequence, more accurate visualizations than the classic MC will be achieved.

3. To have an acceptable computational cost.

To achieve these goals we rely mostly on supplementary points either lying in the faces, or placed in the interior of the cube. No information about neighbour cells is used; neither is it necessary to decompose the cube. These supplementary points are points which lie on the isosurface so satisfying (4.13). The crucial matter is to define and use them wisely.

Before proceeding we would like to draw attention to situations such as the isosurface intersecting corners of the cube or an intersection of the isosurface with a cell face being a degenerate contour (two crossing lines). As particular cases, they should be dealt with accordingly though some of the conclusions we will reach hereafter can be adjusted to those situations.

4.2.1 Supplementary points

In the previous Chapter we have used shoulder points in each 2D cell to achieve a more accurate contour plot. This approach is applicable in the faces of the cubic cell. What then remains to be done is to look at the interior of cell, and find similar points for that purpose. They must be lying on the isosurface. A useful idea is to picture a cubic cell as a range of many 2D slices. That is, an isosurface seen as a pile of consecutive parallel 2D contours. Having that in mind, we set out the role of the supplementary points as follows: in the first place they help to define the correct topological solution. Secondly the

solution obtained is more accurate in comparison to MC. Thirdly they make triangulation more systematic. These points are classed as *face shoulder points*, *inflection points* or *bi-shoulder points*. In the following we define those points, and then we provide a detailed description.

Definition 4.2.1.1 (Face shoulder point). *A face shoulder point is a point on the surface that lies on the face of the cubic cell and is a shoulder point in relation to the contour on that face. It corresponds to the shoulder point discussed in the previous Chapter.*

Definition 4.2.1.2 (Inflection point). *An inflection point is a point on the surface in the interior of the cell such that two first derivatives of* (4.13) *vanish. If the two derivatives are with respect to* x *and* y *, then we denote the inflection point by* $\mathcal{I}z$*. Similarly,* $\mathcal{I}y$ *for* x *and* z *derivatives and* $\mathcal{I}x$ *for* y *and* z *derivatives.*

Definition 4.2.1.3 (Bi-shoulder point). *A bi-shoulder point is a point on the surface in the interior of the cell, such that it is simultaneously a shoulder point on the contour of* (4.13) *on two orthogonal slices through the cube.*

4.2.1.1 Face shoulder points

F of (4.13) is a bilinear interpolant on any face of the cell. This means that all the results in the previous Chapter about contouring are applicable. For each pair of intersection points at the edges of a cell face, we obtain a shoulder point in that face as in contouring. In case of ambiguity we use the asymptotic decider to establish the correct topology in the face. The use of face shoulder points enables a more accurate representation of the intersection of the isosurface with the face. They are described in detail in the previous Chapter, Section 3.3.1 on page 45.

4.2.1.2 Inflection points

These points not only lie on the surface within the cube but also two of the first derivatives vanish there. Therefore a point that satisfies any one of the Equations 4.14, 4.15 or 4.16

is an inflection point.

$$\mathcal{I}z \;:\quad F(x,y,z)=0\,,\qquad F_x(x,y,z)=0\,,\qquad F_y(x,y,z)=0\,. \tag{4.14}$$

$$\mathcal{I}y \;:\quad F(x,y,z)=0\,,\qquad F_x(x,y,z)=0\,,\qquad F_z(x,y,z)=0\,. \tag{4.15}$$

$$\mathcal{I}x \;:\quad F(x,y,z)=0\,,\qquad F_y(x,y,z)=0\,,\qquad F_z(x,y,z)=0\,. \tag{4.16}$$

The greatest feature of a inflection point is that it reflects topology in the interior of the cell. Figure 4.10 attempts to clarify this characteristic. It implies the existence of a degenerate contour on some orthogonal slice through the cube: a contour defined by two straight-lines that intersect at the saddle point.

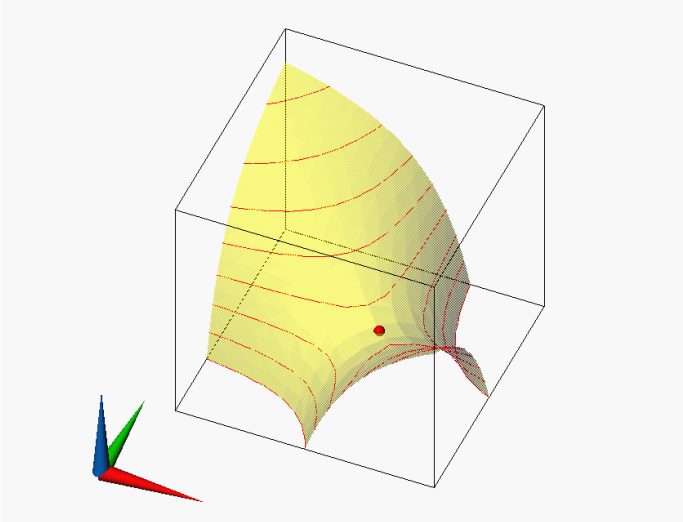


Figure 4.10: Inflection point $\mathcal{I}z = (x_t,\, y_t,\, z_t)$ such that $F = F_x = F_y = 0$. The figure also shows a range of (red) 2D contours parallel to the face $\mathcal{F}_z = 0$ and superimposed on the (yellow but transparent) isosurface (MC configuration 3.b). $\mathcal{I}z$ lies on the horizontal slice $\Omega = z_t$ that behaves as frontier between two different topologies, immediately above and below. In relation to that contour, $(x_t,\, y_t)$ is a saddle point but also part of the contour. Notice that the bottom face $z = 0$ is ambiguous.

Now let us obtain the formulae to compute inflection points. The first derivatives of

F of (4.13) are:

$$F_x = e + g\,y + f\,z + h\,y\,z\,, \tag{4.17}$$

$$F_y = c + g\,x + d\,z + h\,x\,z\,, \tag{4.18}$$

$$F_z = b + f\,x + d\,y + h\,x\,y\,, \tag{4.19}$$

and so

$$a + c\,y + b\,z + d\,y\,z = 0\,, \qquad \text{if } F = 0\,,\ F_x = 0\,. \tag{4.20}$$

$$a + e\,x + b\,z + f\,x\,z = 0\,, \qquad \text{if } F = 0\,,\ F_y = 0\,. \tag{4.21}$$

$$a + e\,x + c\,y + g\,x\,y = 0\,, \qquad \text{if } F = 0\,,\ F_z = 0\,. \tag{4.22}$$

There are up to two solutions to consider for (4.14). They are:

$$x = -\frac{a + b\,z}{e + f\,z} \quad \text{(by 4.21)}\,, \qquad y = -\frac{a + b\,z}{c + d\,z} \quad \text{(by 4.20)}\,, \qquad z = \gamma\,, \tag{4.23}$$

where γ satisfies the second-order equation (by (4.17) setting $F_x = 0$ and (4.20) or else by (4.18) setting $F_y = 0$ and (4.21))

$$(d\,f - b\,h)\,\gamma^2 + (d\,e + c\,f - b\,g - a\,h)\,\gamma + (c\,e - a\,g) = 0\,. \tag{4.24}$$

Similarly, for (4.15)

$$x = -\frac{a + c\,y}{e + g\,y} \quad \text{(by 4.20)}\,, \qquad y = \beta\,, \qquad z = -\frac{a + c\,y}{b + d\,y} \quad \text{(by 4.22)}\,, \tag{4.25}$$

with (by (4.17) and (4.20) or else by (4.19) and (4.22))

$$(d\,g - c\,h)\,\beta^2 + (d\,e + b\,g - c\,f - a\,h)\,\beta + (b\,e - a\,f) = 0\,. \tag{4.26}$$

And finally for (4.16)

$$x = \alpha\,, \qquad y = -\frac{a + e\,x}{c + g\,x} \quad \text{(by 4.22)}\,, \qquad z = -\frac{a + e\,x}{b + f\,x} \quad \text{(by 4.21)}\,, \tag{4.27}$$

with (by (4.18) and (4.21) or else by (4.19) and (4.22))

$$(f\,g - e\,h)\,\alpha^2 + (c\,f + b\,g - d\,e - a\,h)\,\alpha + (b\,c - a\,d) = 0\,. \tag{4.28}$$

Next we draw some relationships among these inflection points, which will be useful in later Sections.

Proposition 4.2.1.4 (Six inflection points). *When six distinct inflection points as solutions of Equations 4.14 ($\mathcal{I}z$), 4.15 ($\mathcal{I}y$) and 4.16 ($\mathcal{I}x$) exist simultaneously, then all are located in the corners of a cuboid. On each face of that cuboid there are three points: one $\mathcal{I}z$, one $\mathcal{I}y$ and one $\mathcal{I}x$. A polyline can be drawn along the edges of the cuboid linking the six points.*

Proof. First let us establish relationships between the different types of inflection points.

$\mathcal{I}z$ and $\mathcal{I}y$. The formulae to compute $\mathcal{I}z$ are given by (4.23); the y coordinate derives from (4.20) and the z coordinate can be derived from both (4.17) and (4.20). The formulae to compute $\mathcal{I}y$ are given by (4.25); in this case the y coordinate can be derived from both (4.17) and (4.20), and the z coordinate derives from (4.20). Therefore a $(y,\,z)$ solution for $\mathcal{I}z$ is also a $(y,\,z)$ solution for $\mathcal{I}y$ and vice-versa. Notice that the formulae for $\mathcal{I}z$ give the x coordinate depending on z and for $\mathcal{I}y$ depending on y.

$\mathcal{I}z$ and $\mathcal{I}x$. From (4.23) we see that the x coordinate of $\mathcal{I}z$ derives from (4.21) and the z coordinate can be derived from both (4.18) and (4.21). The formulae to compute $\mathcal{I}x$ are given by (4.27); in this case the x coordinate can be derived from both (4.18) and (4.21), and the z coordinate derives from (4.21). And looking also at the formulae for the y coordinate we conclude a $(x,\,z)$ solution for $\mathcal{I}z$ is also a $(x,\,z)$ solution for $\mathcal{I}y$ and vice-versa.

$\mathcal{I}y$ and $\mathcal{I}x$. From (4.25) we see that the x coordinate of $\mathcal{I}y$ derives from (4.22) and the y coordinate can be derived from both (4.19) and (4.22). From (4.27), the x coordinate of $\mathcal{I}x$ can be derived from both (4.19) and (4.22), and the y coordinate derives from (4.22). Also from the formulae, for $\mathcal{I}y$ the z coordinate depends on y and for $\mathcal{I}x$ depends on x. Therefore a $(x,\, y)$ solution for $\mathcal{I}y$ is also a $(x,\, y)$ solution for $\mathcal{I}x$ and vice-versa.

Now we will use a constructive proof. We have six distinct points: two $\mathcal{I}z$, two $\mathcal{I}y$ and two $\mathcal{I}x$, defined by the Equations 4.23, 4.25, and 4.27 respectively. Suppose we have computed the two $\mathcal{I}z$, *i.e.*

$$\mathcal{I}z: \qquad (\,\widehat{x}_1,\,\widehat{y}_1,\,\widehat{z}_1\,)\,, \qquad\qquad (\,\widehat{x}_2,\,\widehat{y}_2,\,\widehat{z}_2\,)\,.$$

Due to the relationships between $\mathcal{I}z$ and $\mathcal{I}y$ and between $\mathcal{I}z$ and $\mathcal{I}x$, the other four points will have to be such as

$$\mathcal{I}y: \qquad (\,\overset{*}{x}_1,\,\widehat{y}_1,\,\widehat{z}_1\,)\,, \qquad\qquad (\,\overset{*}{x}_2,\,\widehat{y}_2,\,\widehat{z}_2\,)\,.$$
$$\mathcal{I}x: \qquad (\,\widehat{x}_1,\,\overline{y}_1,\,\widehat{z}_1\,)\,, \qquad\qquad (\,\widehat{x}_1,\,\overline{y}_2,\,\widehat{z}_2\,)\,.$$

But taking also into account the relationship between $\mathcal{I}y$ and $\mathcal{I}x$, and knowing that there are exactly six points, we conclude that they have to be as

$$\mathcal{I}z: \qquad (\,\widehat{x}_1,\,\widehat{y}_1,\,\widehat{z}_1\,)\,, \qquad\qquad (\,\widehat{x}_2,\,\widehat{y}_2,\,\widehat{z}_2\,)\,.$$
$$\mathcal{I}y: \qquad (\,\widehat{x}_2,\,\widehat{y}_1,\,\widehat{z}_1\,)\,, \qquad\qquad (\,\widehat{x}_1,\,\widehat{y}_2,\,\widehat{z}_2\,)\,.$$
$$\mathcal{I}x: \qquad (\,\widehat{x}_1,\,\widehat{y}_2,\,\widehat{z}_1\,)\,, \qquad\qquad (\,\widehat{x}_1,\,\widehat{y}_1,\,\widehat{z}_2\,)\,.$$

There are just two values for each of x, y and z. The $\mathcal{I}z$ are end-points of a diagonal of the cuboid. The same is true for $\mathcal{I}y$ and for $\mathcal{I}x$. In each face of the cuboid we have one $\mathcal{I}z$, one $\mathcal{I}y$ and one $\mathcal{I}x$. Consequently we can draw a polyline along edges of the cuboid linking the six inflection points. Figure 4.11 shows a practical example from MC configuration 4, with positive vertices as 000 and 111. □

xyz	F_{xyz}
000	5.0
001	-1.5
010	-2.0
011	-3.0
100	-3.0
101	-2.0
110	-1.0
111	15.0

Cuboid	
x :	0.216 to 0.583
y :	0.193 to 0.648
z :	0.159 to 0.671

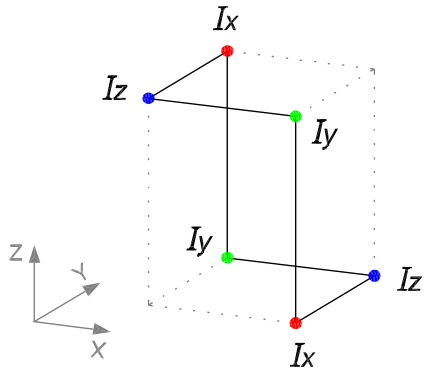


Figure 4.11: Six inflection points located on the corners of a cuboid. This is MC configuration 4, with the positive vertices as 000 and 111 (data values are also of Figure 4.6(b)). On each face there is one $\mathcal{I}z$, one $\mathcal{I}y$ and one $\mathcal{I}x$. Inflections of the same type are diagonnally opposite. So we can link all these points by a polyline along the edges of the cuboid.

Proposition 4.2.1.5 (Same discriminant to compute inflection points). *The three second order Equations 4.24, 4.26, and 4.28, to be solved in order to obtain the inflection points, respectively $\mathcal{I}z$, $\mathcal{I}y$ and $\mathcal{I}x$, have the same discriminant.*

Proof. The discriminants for the three Equations are:

$$\triangle_z = (de + cf - bg - ah)^2 - 4\,(df - bh)\,(ce - ag)\,, \qquad \text{by (4.24)}\,. \tag{4.29}$$

$$\triangle_y = (de + bg - cf - ah)^2 - 4\,(dg - ch)\,(be - af)\,, \qquad \text{by (4.26)}\,. \tag{4.30}$$

$$\triangle_x = (cf + bg - de - ah)^2 - 4\,(fg - eh)\,(bc - ad)\,, \qquad \text{by (4.28)}\,. \tag{4.31}$$

From (4.29) we obtain:

$$\begin{aligned}
\triangle_z &= (de)^2 + (cf)^2 + (bg)^2 + (ah)^2 \\
&\quad + 2\,(decf - debg - deah - cfbg - cfah + bgah) \\
&\quad - 4\,(dfce - dfag - bhce + bhag) \\
&= (de)^2 + (cf)^2 + (bg)^2 + (ah)^2 \\
&\quad - 2\,(decf + debg + deah + cfbg + cfah) \\
&\quad + 4\,(dfag + bhce)\,.
\end{aligned} \tag{4.32}$$

And from (4.30):

$$\begin{aligned}\triangle_y &= (de)^2 + (bg)^2 + (cf)^2 + (ah)^2 \\ &\quad + 2\,(debg - decf - deah - bgcf - bgah + cfah) \\ &\quad - 4\,(dgbe - dgaf - chbe + chaf) \\ &= (de)^2 + (bg)^2 + (cf)^2 + (ah)^2 \\ &\quad - 2\,(debg + decf + deah + bgcf + bgah + cfah) \\ &\quad + 4\,(dgaf + chbe)\,.\end{aligned} \tag{4.33}$$

Finally from (4.31):

$$\begin{aligned}\triangle_z &= (cf)^2 + (bg)^2 + (de)^2 + (ah)^2 \\ &\quad + 2\,(cfbg - cfde - cfah - bgde - bgah + deah) \\ &\quad - 4\,(fgbc - fgad - ehbc + ehad) \\ &= (cf)^2 + (bg)^2 + (de)^2 + (ah)^2 \\ &\quad - 2\,(cfbg + cfde + cfah + bgde + bgah + deah) \\ &\quad + 4\,(fgad + ehbc)\,.\end{aligned} \tag{4.34}$$

One can verify that Equations 4.32, 4.33 and 4.34 are the same: $\triangle_z = \triangle_y = \triangle_x$. □

4.2.1.3 Bi-shoulder points

The bi-shoulder points are the 3D extension of face shoulder points. We will rely on Figures 4.12, 4.13 and 4.14 to illustrate the idea. First, let us consider an orthogonal slice through the cube in Figure 4.12, say of y equal to a constant value. It is possible to define a shoulder point in the contour on that slice. Now if we conceive sweeping across the $y-$ direction, defining a range of slices, we establish a curve as the locus of those shoulder points. Proceeding exactly in the same way but now in another direction, say $z-$, we obtain a different curve. An intersection of both matches the definition of a bi-shoulder point.

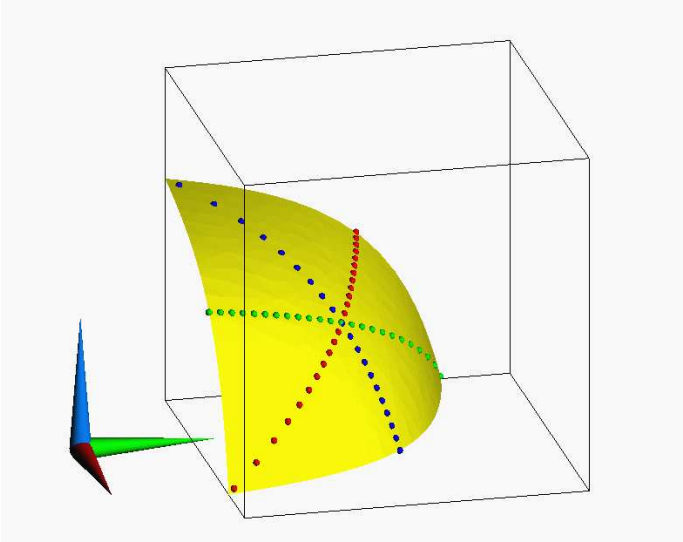


Figure 4.12: Bi-shoulder point in MC configuration 1. The three coloured curves on the isosurface are the locus of many shoulder points on different slices: red if considering slices of x constant, green for slices of y constant and blue for z constant. These curves intersect to each other just once and at the same point.

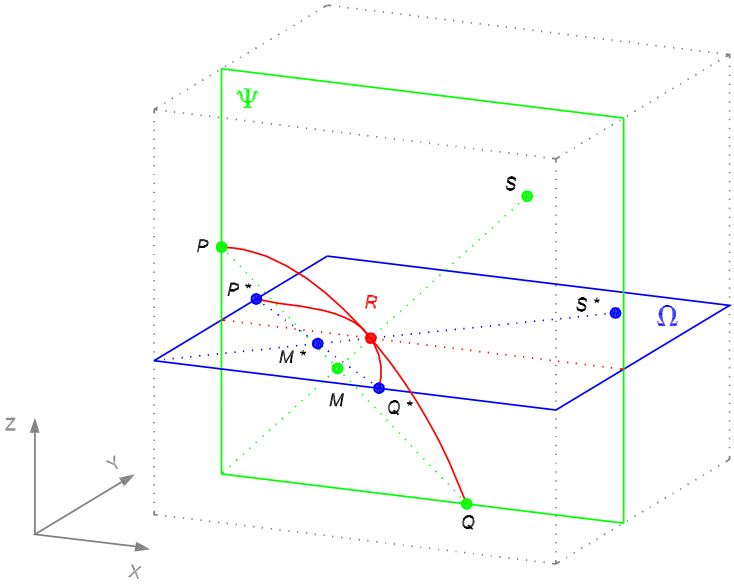
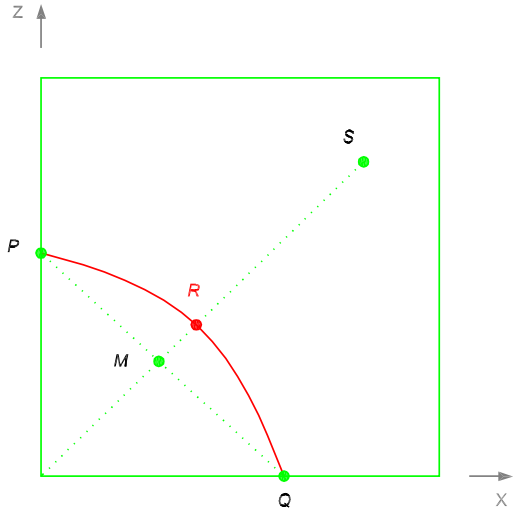


Figure 4.13: Definition of a bi-shoulder point, $R = (x_t, y_t, z_t)$. It is simultaneously a shoulder point for the contours drawn on each of the orthogonal slices, $\Psi = y_t$ and $\Omega = z_t$. P and Q are intersection points, S saddle points and M middle points between P and Q.



Slice $\Psi = y_t$

$$F = \psi_0\, x + \psi_1\, z + \psi_2\, x\, z + \psi_3$$

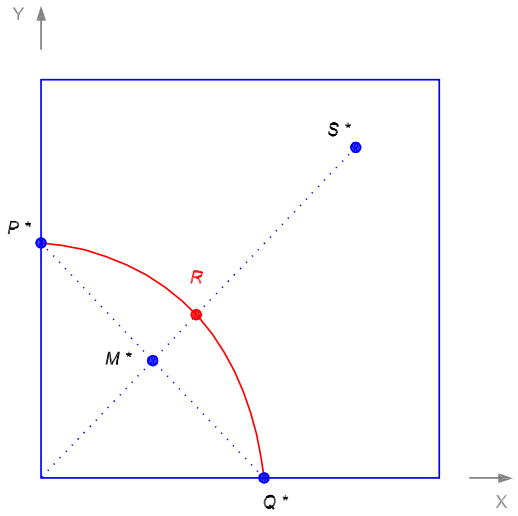
$$\psi_0 = e + g\, y_t\,, \quad \psi_1 = b + d\, y_t\,,$$

$$\psi_2 = f + h\, y_t\,, \quad \psi_3 = a + c\, y_t\,.$$

$$P = (\,0\,,\ y_t\,,\ -\psi_3/\psi_1\,)$$

$$Q = (\,-\psi_3/\psi_0\,,\ y_t\,,\ 0\,)$$

$$S = (\,-\psi_1/\psi_2\,,\ y_t\,,\ -\psi_0/\psi_2\,)$$



Slice $\Omega = z_t$

$$F = \omega_0\, x + \omega_1\, y + \omega_2\, x\, y + \omega_3$$

$$\omega_0 = e + g\, z_t\,, \quad \omega_1 = c + d\, z_t\,,$$

$$\omega_2 = g + h\, z_t\,, \quad \omega_3 = a + b\, z_t\,.$$

$$P^* = (\,0\,,\ -\omega_3/\omega_1\,,\ z_t\,)$$

$$Q^* = (\,-\omega_3/\omega_0\,,\ 0\,,\ z_t\,)$$

$$S^* = (\,-\omega_1/\omega_2\,,\ -\omega_0/\omega_2\,,\ z_t\,)$$

Figure 4.14: Contours on the orthogonal slices Ψ and Ω depicted in Figure 4.13.

In the example of Figure 4.12 — the simple MC configuration 1, with the positive vertex at the origin of the coordinate system — the unique bi-shoulder can be computed as the intersection of three surfaces: the isosurface, described by F of (4.13) and two more surfaces. Figures 4.13 and 4.14 help to understand these two additional surfaces. In the top diagram of Figure 4.14, notice that the shoulder point R, on the $y-$ slice shown, lies on the line joining the origin and the saddle point S (passing the middle point M). Imagine now sweeping through the cube from $y = 0$ to $y = 1$; this line — that changes from slice to slice — will sweep out a surface. Similarly, in the lower diagram, sweeping the line from the origin to the saddle point S^* will sweep out another surface. The intersection of these two surfaces with the isosurface will give the bi-shoulder point. The two surfaces are defined by (see Figure 4.14)

$$z \;=\; x\;\frac{e+g\,y}{b+d\,y}\,, \qquad \text{by sweeping across the } y- \text{ direction (slices } \Psi = y_t)\;. \tag{4.35}$$

$$y \;=\; x\;\frac{e+f\,z}{c+d\,z}\,, \qquad \text{by sweeping across the } z- \text{ direction (slices } \Omega = z_t)\;. \tag{4.36}$$

In this simple case (MC configuration 1), these formulae are easily established. Recall that by definition a face shoulder point R lies on the line joining the middle point M and the saddle point S. Fortunately in the case above, the origin, M and S are collinear which yields the simple formulae. However, the system can not be solved analytically. An attempt to solve it leads to the relation $b\,z - c\,y + f\,x\,z - g\,x\,y = 0$. That turns out to be the equation we would have obtained if we had considered a sweeping across the $x-$ direction. So regardless of the two directions chosen the result is the same. The solution is such that $x\,F_x = y\,F_y = z\,F_z$.

In general, the formulae are more complex. Once the formulae is established we can always use numerical routines such as the ones available in the NAG library to solve the non-linear equations. We should be aware however that, unlike the example above (isosurface intersecting only three cell faces), there are many situations where more than one bi-shoulder point can be obtained. We will show pictures depicting some cases.

In any case, we provide a general but simple procedure to compute a bi-shoulder

point, which has worked well in practice. The method is as follows: first, we decide from which faces the sweeping will take place. For a particular topological polygon, we use two adjacent faces upon which the polygon lies. For the sake of clarity let us consider the faces $\mathcal{F}y$ and $\mathcal{F}z$, with coordinates y and z constants, respectively. At any step in the algorithm there are two orthogonal slices defined, Ψ and Ω, parallel to $\mathcal{F}y$ and $\mathcal{F}z$, respectively. Suppose that at the step t we have the slice $\Psi = y_t$. The shoulder point on Ψ is computed, say $R_t = (x_t,\, y_t,\, z_t)$. The coordinate value z_t is then used to define the slice $\Omega = z_t$. Again a shoulder point on the contour of that slice is computed, $R_t^* = (x_t^*,\, y_t^*,\, z_t^*)$. If y_t and y_t^* are close enough to be considered as the same (dependent on the precision established for the numerical approximation) then both R_t and R_t^* are considered as the same point; the bi-shoulder point is found. If not, there it follows a next iteration t_{i+1}. The guess for the next slice Ψ is the average of y_t and y_t^*.

The first slice Ψ is defined by the y coordinate of the face shoulder point in the counterpart face $\mathcal{F}z$. In the end we aim to move from the faces towards the interior of the cell.

To summarise this Section, we have introduced in detail the three types of supplementary points — face shoulder points, inflection points and bi-shoulder points — that will be the foundation of our accurate MC. Next we will discuss how to accomplish the triangulation.

4.2.2 Triangulation

Although it is crucial to have correct topologies in the faces, this by itself is far from reaching the goals set. Triangulation as a final step must be carried out carefully otherwise it jeopardizes previous efforts towards accurate solutions.

The methodology is as follows: In the first place, we establish the topological polygons as the MC does, using the asymptotic decider. By doing so we achieve the correct topology in the faces and so positional continuity C_0 between cells. Then we extend the topological polygons to include face shoulder points. To recall, for each pair of adjacent intersection points we define a 2D shoulder point. The boundary of any *extended topolog-*

ical polygon (see complete definition below) will again lie on cell faces. Then to proceed with the triangulation we use point(s) in the interior of the cell: inflection point(s) or else a bi-shoulder point.

Definition 4.2.2.1 (Extended topological polygon). *An extended topological polygon P^E is a topological polygon as described in Definition 4.1.1.3 on page 72 but including the respective face shoulder point between each pair of intersection points. The notation for the classification is exactly the same but changing P for P^E. The definition of equivalence still holds, i.e. they are equivalent if the notation is the same.*

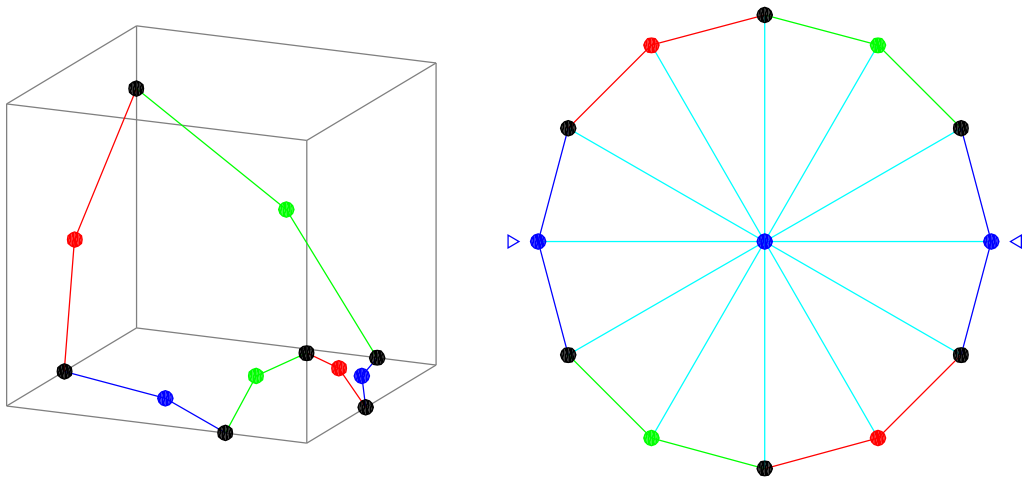
As can be concluded from above, our triangulation approach is driven by the topological polygons rather than the MC configuration. This is quite the opposite to the traditional MC. Indeed, the extended polygons are always triangulated one by one, regardless of the MC configuration from which they arose. Nevertheless, we would like to draw attention to the special case of tunnels, that follows an approach on its own.

In the next three Sections, 4.2.2.1, 4.2.2.2, and 4.2.2.3, we will give detailed information about triangulation. The first Section is concerned with polygons that lie upon at least one loop-back face. In the second Section we consider the case with no loop-back faces at all. Tunnels are discussed in the third Section.

For illustrative purposes we use coloured diagrams in these Sections. Consider Figure 4.15. First, there is the closed boundary representing the extended polygon that lies on the cell faces. We have there the intersection points, the face shoulder points and edges linking those. As far as colouring is concerned, intersection points are coloured black, whereas red, green and blue are used to colour the remaining entities in the boundary. Each of these three colours is assigned to a pair of faces, one colour to faces $x = 0$, $x = 1$, another one to $y = 0$, $y = 1$ and the last one to $z = 0$, $z = 1$. If six colours were adopted that would have cluttered the diagrams so much that information would have not been conveyed properly. Besides, there is no real need to emphasise the difference of faces within each of the pairs mentioned.

The interior of the surface is ultimately described by cyan edges. The vertices of triangles are two neighbouring boundary points and a point in the interior. If there are

no loop-back faces that point is a bi-shoulder point — coloured with magenta. Otherwise the points in the interior are inflection points: between one to three when there is no tunnel; six in the case of a tunnel. The colours used for inflection points are the same as the related face shoulder points (red, green or blue). Figure 4.15 gives an example for a polygon of six intersection points and one loop-back face, ${P^E}_{6/1}$. Figure 4.15(a) shows the topological polygon within the cube; Figure 4.15(b) is a convenient way of visualizing the triangulation: the 3D topological polygon is flattered into a regular planar polygon whose points correspond in order to the points of the original 3D polygon. The presence of a loop-back face is indicated by small triangles against the corresponding face shoulder points.



(a) (b)

Figure 4.15: Diagram showing triangulation of ${P^E}_{6/1}$. This is an example of MC configuration 3.b, already shown in Figure 4.10. On (a) we present the polygon upon the cube. The diagram is on (b). The boundary comprises the faces: $z = 0$ (bottom), $y = 0$ (front), $x = 1$ (right) and so on. Each of x, y, z corresponds to one colour: (red, green and blue respectively). The intersection points are black spheres and the face shoulder points are in between. These are coloured as the faces upon which they lie. The inflection point in the interior inherits the colour of the related face shoulder points. The triangulation is completed by cyan lines between each point in the boundary and the point in the interior. Finally there are two small triangles indicating the loop-back face, and coloured accordingly.

Several representations of isosurfaces within the cubic cell are shown. Our bench-

mark is the exact trilinear interpolant function F of (4.13), and so we show the underlying isosurface of F to compare results. We use images obtained when F of (4.13) is super-sampled and then the isosurface from the new data set is extracted. The sampling factor used is twenty, leading to a data set of dimension $[20 \times 20 \times 20]$, in the same interval $[0,\ 1]$. We have deliberately not attempted to shade the surface in a photo-realistic way. Indeed, each triangle has got constant colour, because we want to emphasise how the triangulation is done.

4.2.2.1 Polygons on loop-back faces

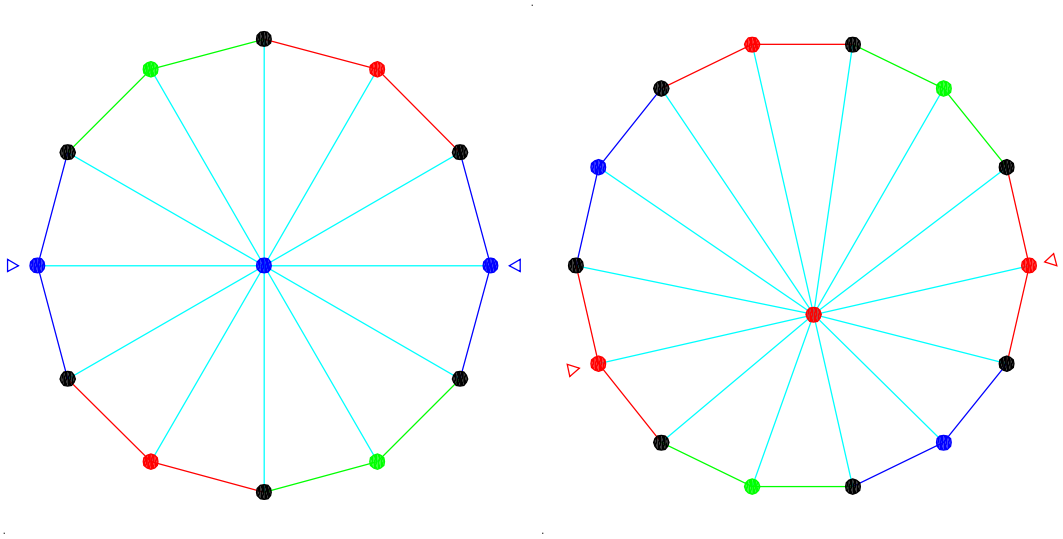
All the topological polygons P^E that lie upon loop-back face(s) are indicated in Table 4.2.

Polygons P^E	MC configurations
${P^E}_{6/1}$	3.b, 7.b, 13.bc
${P^E}_{7/1}$	6.b
${P^E}_{8/1}$	10.bd
${P^E}_{8/2}$	12.bd
${P^E}_{9/3}$	7.c, 13.de
${P^E}_{12/3}$	13.i

Table 4.2: Topological polygons P^E lying on loop-back faces and in which MC configurations they may appear (see also Table 4.1). The notation is according to Definition 4.2.2.1. The triangulations proposed are shown in Figures 4.16, 4.17 and 4.18.

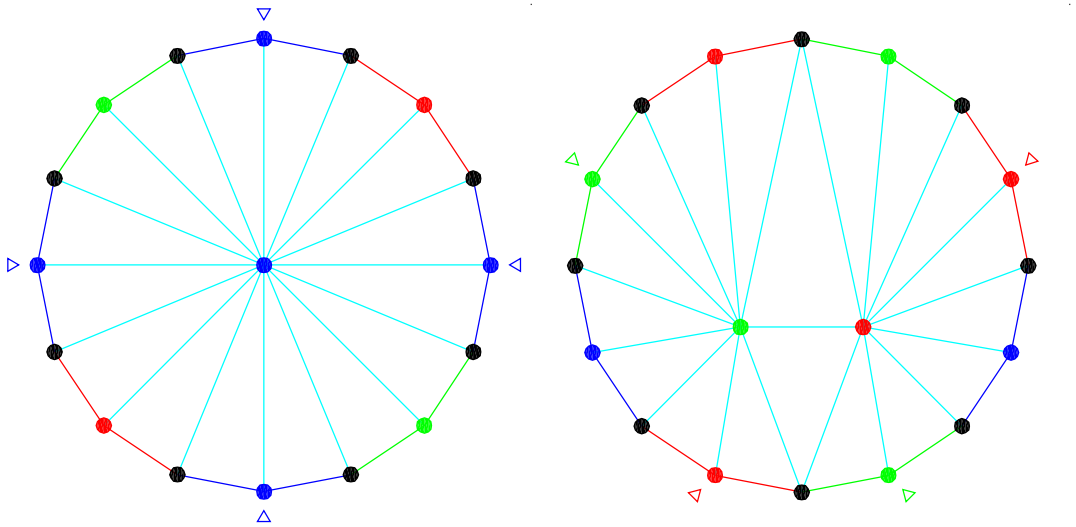
Two major but complementary factors determine the way triangulation is done. Firstly, to ensure correct topology and continuity, no in-face triangles are allowed. Particular care is needed to avoid this on loop-back faces where there are four intersection points and a pair of face shoulder points. Secondly, a loop-back face implies the existence of a inflection point. Looking back at the characterisation of inflection points, we conclude they are suitable to be used here. They lie in the interior of the cube so we will able to avoid in-face triangles. Moreover, if a polygon lies on a loop-back face, say $\mathcal{F}_{z=0}$, then the inflection point $\mathcal{I}z$ (point on the surface such that both first derivatives with respect to x and y vanish) will reference a slice parallel to $\mathcal{F}_{z=0}$ such that the contour drawn on it is degenerate. In the slices immediately below and above, the contours show opposite shape (see earlier Figure 4.10). Thus an inflection point is a critical point.

In the following Figures 4.16, 4.17 and 4.18, we propose the triangulation for each of these polygons P^E in Table 4.2. Then there follows some examples of isosurfaces within a cubic cell: Figures 4.19 to 4.24.



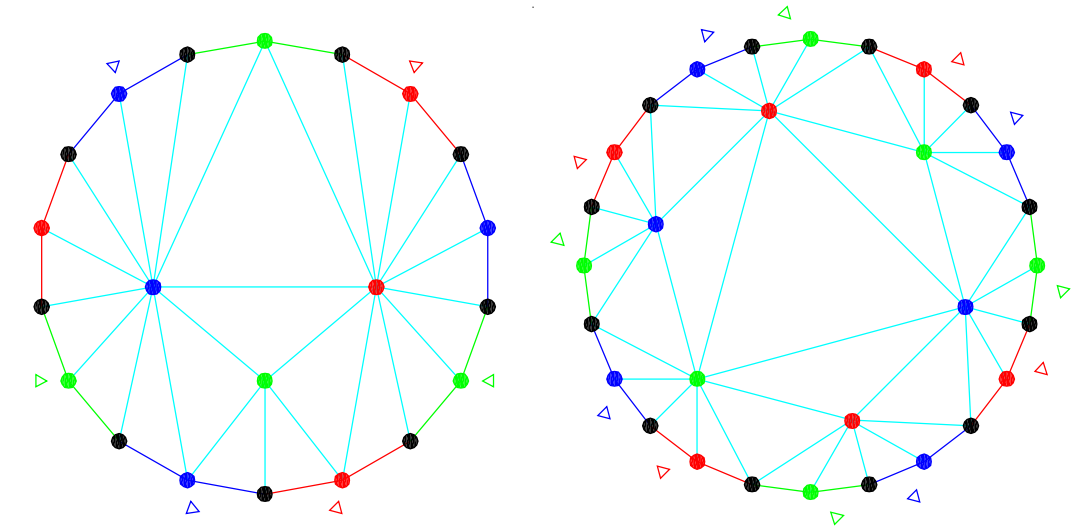
(a) $P^E{}_{6/1}$ (b) $P^E{}_{7/1}$

Figure 4.16: Triangulation of polygons P^E that lie on one loop-back face. On (a) there are six intersection points and five faces. On (b) there are seven intersection points and six faces.



(a) $P^E{}_{8/1}$ (b) $P^E{}_{8/2}$

Figure 4.17: Triangulation of polygons P^E with eight intersection points, lying on six faces with two of them being loop-back faces. On (a) the two loop-back faces are opposite to each other, whereas on (b) they are not.



(a) $P^E{}_{9/3}$ (b) $P^E{}_{12/3}$

Figure 4.18: Triangulation of polygons P^E that lie on loop-back faces. On (a) there are nine intersection points, lying on all the six faces of the cube but three of them are loop-back faces. On (b) all the twelve edges of the cube are intersected so all the faces are loop-back faces. We should point out that the interior area of the inflection points can be triangulated in different order.

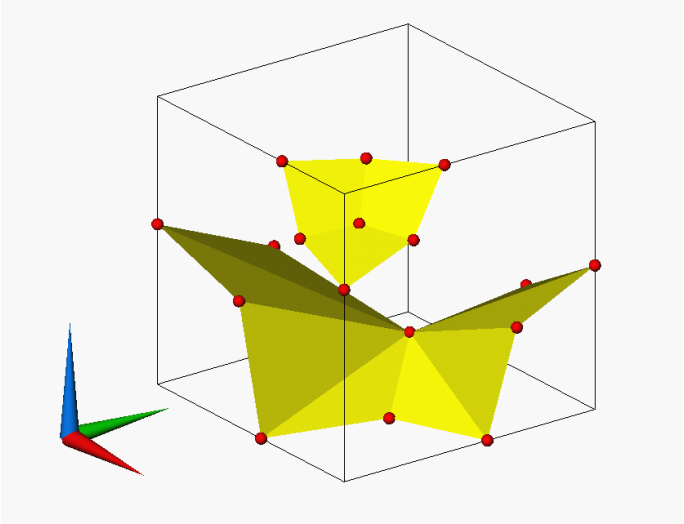
Topological polygon
$P^E{}_{6/1}$

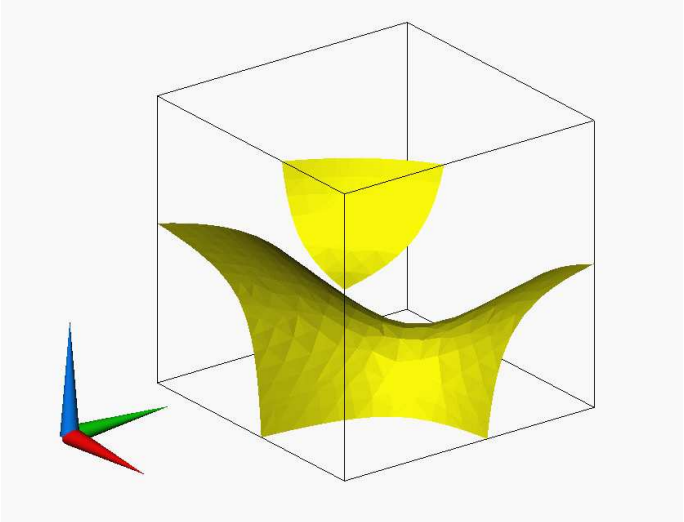
MC configuration: 7.b

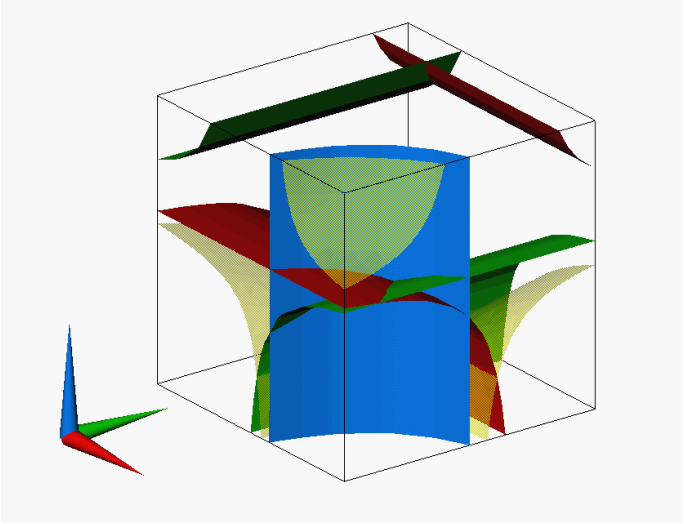
xyz	F_{xyz}
000	2.5
001	-2.0
010	-1.0
011	-1.0
100	-2.0
101	1.0
110	1.5
111	-1.5

Inflection point	
$\mathcal{I}z$	(0.532, 0.611, 0.207)

Body saddle point	F
(0.778, 0.364, 0.531)	-0.293







Pictures.
Our representation is on top whereas the ”exact” isosurface of F is in the middle. The surfaces where the three first derivatives F_x (red) F_y (green) and F_z (blue) vanish are represented at the bottom, as well as the ”exact” surface (yellow but transparent).

Figure 4.19: An example of $P^E{}_{6/1}$.

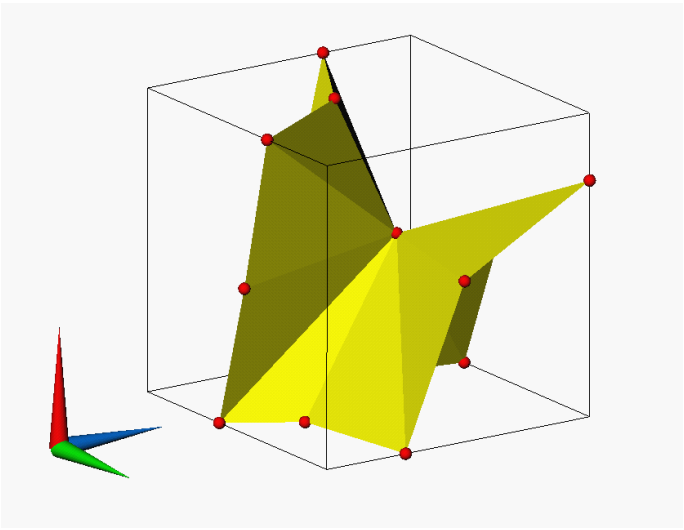
Topological polygon
$P^E{}_{7/1}$

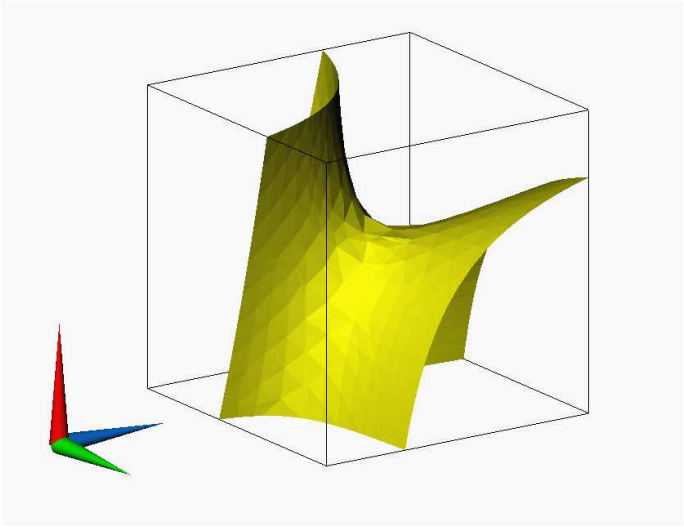
MC configuration: 6.b

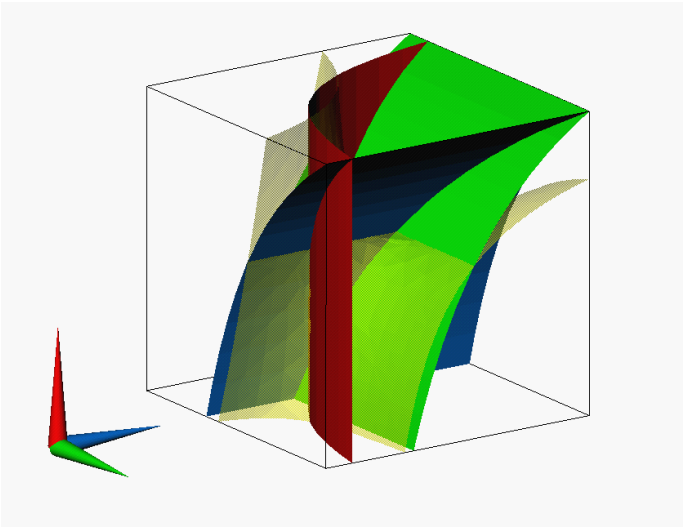
xyz	F_{xyz}
000	1.0
001	-1.5
010	-1.5
011	3.5
100	2.0
101	-1.0
110	-1.0
111	-1.0

Inflection points	
$\mathcal{I}z$	(0.568, 0.237, 0.563)
$\mathcal{I}y$	(0.568, 0.563, 0.237)
$\mathcal{I}x$	(0.568, 0.563, 0.563)

Body saddle point	F
(0.137, 0.373, 0.373)	0.179







Pictures.
Our representation is on top whereas the ”exact” isosurface of F is in the middle. The surfaces where the three first derivatives F_x (red) F_y (green) and F_z (blue) vanish are represented at the bottom, as well as the ”exact” surface (yellow but transparent).

Figure 4.20: An example of $P^E{}_{7/1}$. Although there is just one ambiguous face there is more than one inflection point. In this configuration this number can vary from case to case.

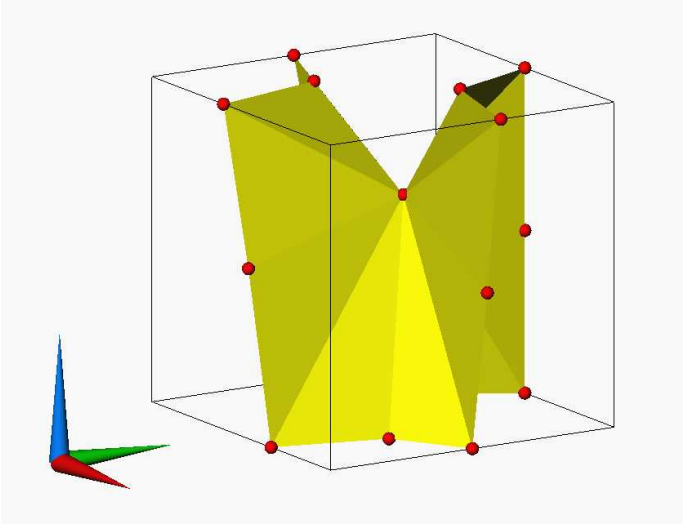
Topological polygon
$P^E{}_{8/1}$

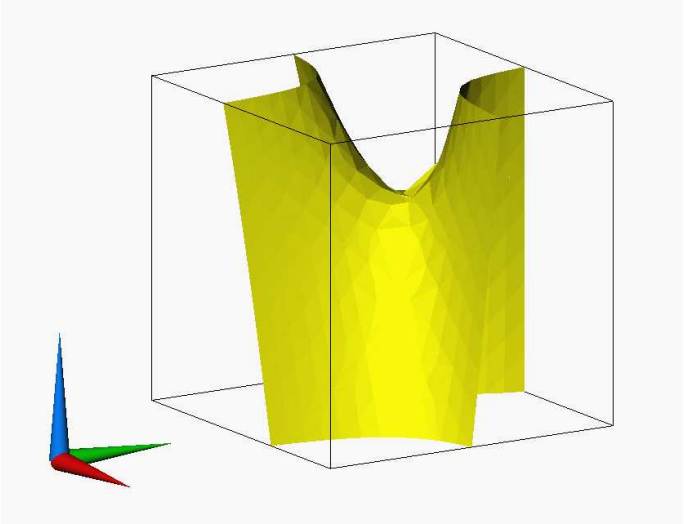
MC configuration: 10.bd

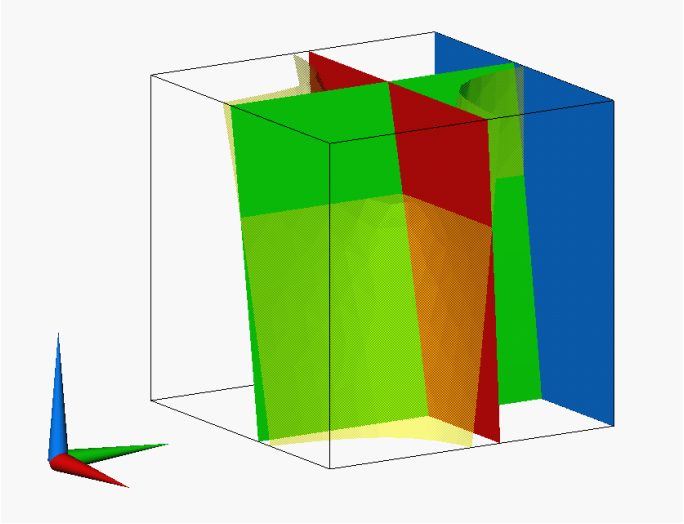
xyz	F_{xyz}
000	2.0
001	1.0
010	-1.0
011	-1.0
100	-1.0
101	-1.5
110	1.0
111	1.0

Inflection point	
$\mathcal{I}z$	$(0.500, 0.571, 0.667)$

Body saddle point
none







Pictures.
Our representation is on top whereas the ”exact” isosurface of F is in the middle. The surfaces where the three first derivatives F_x (red) F_y (green) and F_z (blue) vanish are represented at the bottom, as well as the ”exact” surface (yellow but transparent).

Figure 4.21: An example of $P^E{}_{8/1}$. There is just one inflection point but two ambiguous faces.

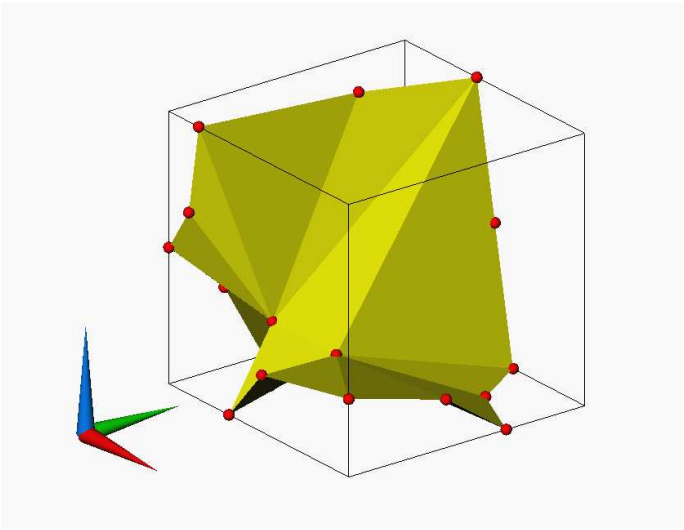
Topological polygon
$P^E{}_{8/2}$

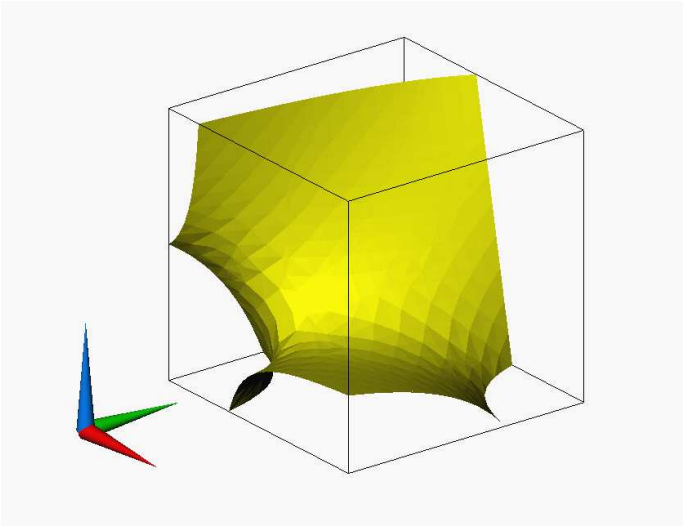
MC configuration: 12.bd

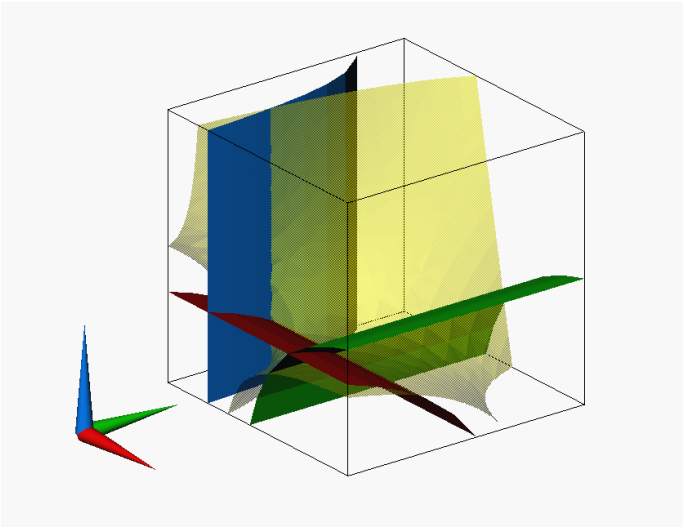
xyz	F_{xyz}
000	1.0
001	-1.0
010	-1.5
011	-1.0
100	-2.0
101	5.0
110	1.0
111	1.5

Inflection points	
$\mathcal{I}z$	(0.556, 0.286, 0.222)
$\mathcal{I}y$	(0.200, 0.286, 0.222)

Body saddle point
none







Pictures.
Our representation is on top whereas the ”exact” isosurface of F is in the middle. The surfaces where the three first derivatives F_x (red) F_y (green) and F_z (blue) vanish are represented at the bottom, as well as the ”exact” surface (yellow but transparent).

Figure 4.22: An example of $P^E{}_{8/2}$.

Topological polygon
$P^E{}_{9/3}$

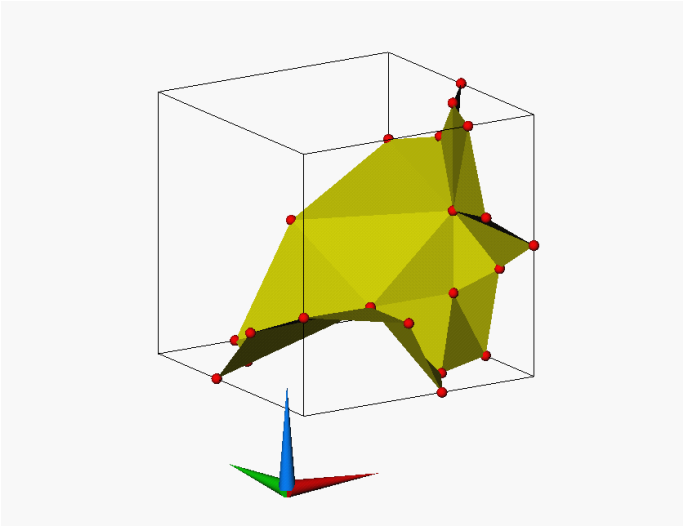
MC configuration: 7.c

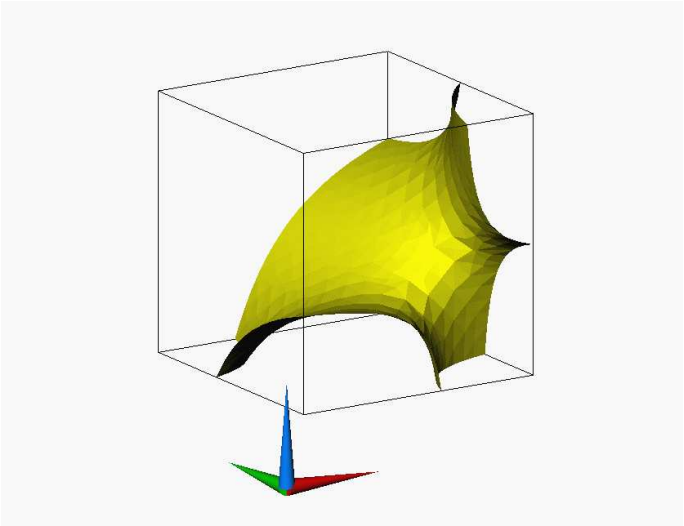
xyz	F_{xyz}
000	1.5
001	-2.5
010	-1.0
011	-1.0
100	-1.0
101	1.0
110	2.0
111	-1.0

Inflection points	
$\mathcal{I}z$	$(0.463, 0.275, 0.280)$
$\mathcal{I}y$	$(0.823, 0.275, 0.280)$
$\mathcal{I}x$	$(0.823, 0.275, 0.595)$

Body saddle point
none

Pictures.
Our representation is on top whereas the ”exact” isosurface of F is in the middle. The surfaces where the three first derivatives F_x (red) F_y (green) and F_z (blue) vanish are represented at the bottom, as well as the ”exact” surface (yellow but transparent).





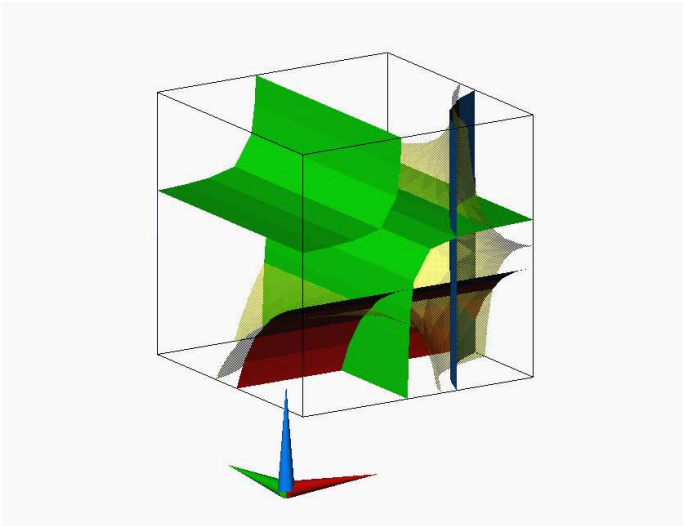


Figure 4.23: An example of $P^E{}_{9/3}$. Note that the three inflection points lie on the same plane (with $y = 0.275$).

Topological polygon
$P^E{}_{12/3}$

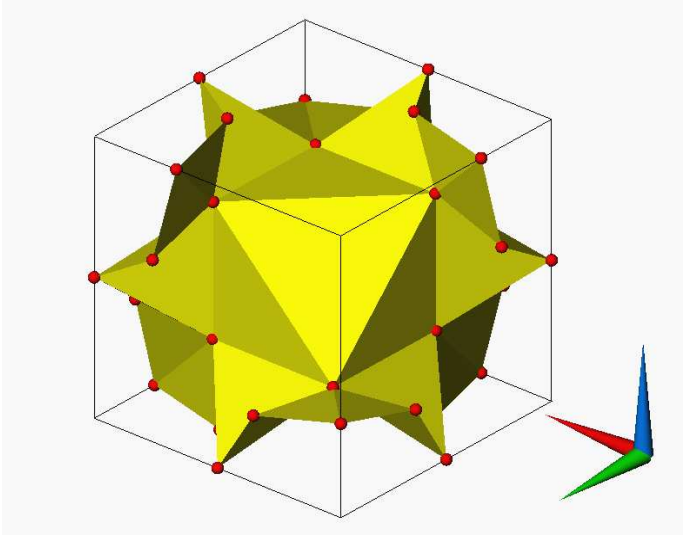
MC configuration: 13.i

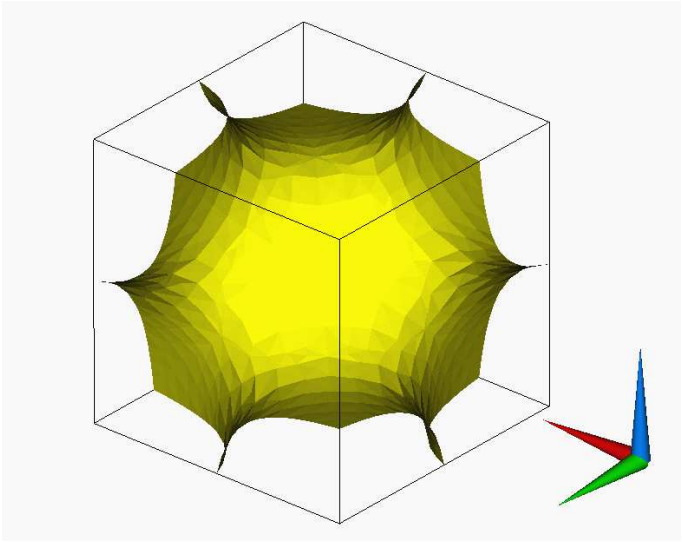
xyz	F_{xyz}
000	1.0
001	-1.0
010	-1.0
011	2.0
100	-2.5
101	1.0
110	1.0
111	-1.0

Inflection points	
$\mathcal{I}z$	(0.2183, 0.7817, 0.2952)
	(0.7048, 0.2952, 0.7817)
$\mathcal{I}y$	(0.7048, 0.7817, 0.2952)
	(0.2183, 0.2952, 0.7817)
$\mathcal{I}x$	(0.7048, 0.7817, 0.7817)
	(0.2183, 0.2952, 0.2952)

Body saddle point
none

Pictures.
Our representation is on top whereas the ”exact” isosurface of F is in the middle. The surfaces where the three first derivatives F_x (red) F_y (green) and F_z (blue) vanish are represented at the bottom, as well as the ”exact” surface (yellow but transparent).





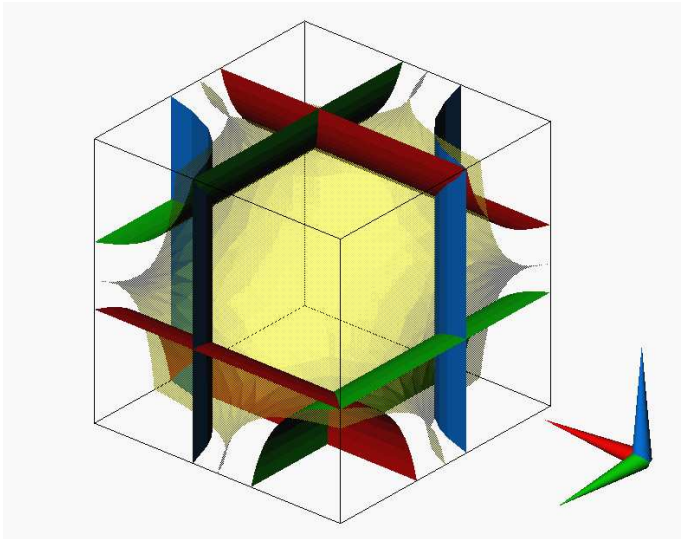


Figure 4.24: An example of $P^E{}_{12/3}$. Note that though there are six inflection points there is no body saddle point.

4.2.2.2 Polygons with no loop-back faces

The topological polygons ${P^E}_{i/0}$ that do not lie on loop-back faces are indicated in Table 4.2.2.2.

Polygons P^E	MC configurations
${P^E}_{3/0}$	1 and 3.a, 6.a, 7.a, 7.b, 7.d, 12.ac, 13.a, 13.bc, 13.de, 13.j
${P^E}_{4/0}$	2, 8 and 6.a, 10.ac
${P^E}_{5/0}$	5 and 12.ac
${P^E}_{6/0}$	9, 11 and 7.d, 13.j

Table 4.3: Topological P^E polygons that do not lie on loop-back faces and in which MC configurations they might appear (see also Table 4.1). The notation is according to Definition 4.2.2.1 and Figure 4.25 shows the triangulation proposed.

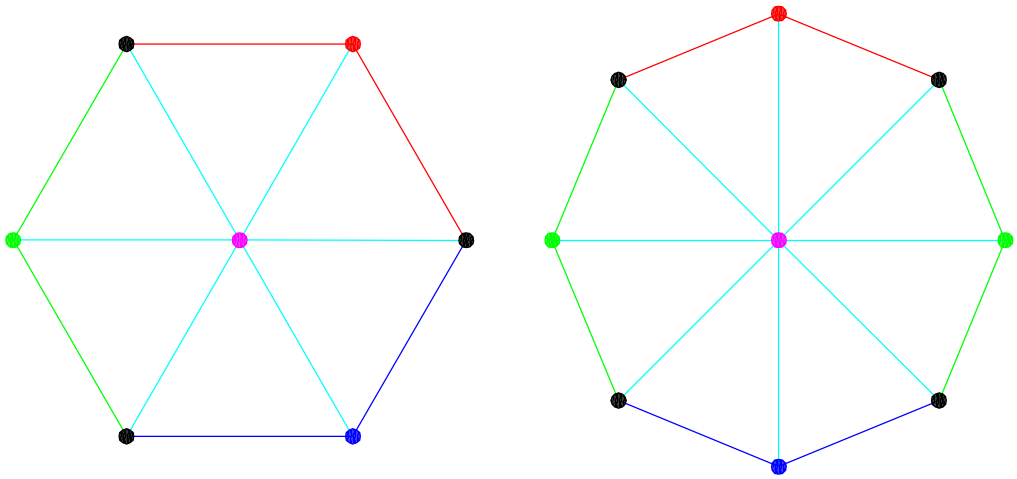
The key point here is that there are no loop-back faces and so no inflection points. In other words, there are no degenerate contours on any orthogonal slice through the cube. Triangulation here is more a matter of accuracy rather than topology. But still it requires at least one internal point. That point will be a bi-shoulder point.

One problem though is to choose from which faces the sweeping takes place to compute the bi-shoulder point. In other words to choose which curves of shoulder points are to be considered. The simple case of ${P^E}_{3/0}$ yields only one solution for the bi-shoulder point, as shown earlier in Figure 4.12. But that is not the general case. As we shall see in following pictures many different bi-shoulder points can be obtained. It is possible to use all of these points and thereby obtain a highly accurate triangulation. However the triangulation becomes complex. We have found the following compromise works well in practice. We use only one bi-shoulder point, but choose it carefully according to the following rule:

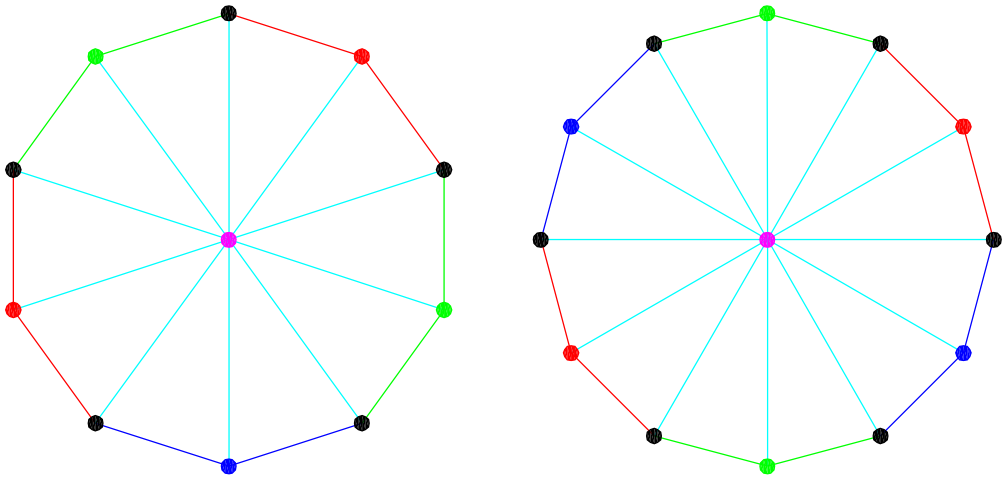
> Consider the set of faces $\{\mathcal{F}_1, \mathcal{F}_2, \cdots \mathcal{F}i\}$ on which the polygon ${P^E}_{i/0}$ lies. Choose $\mathcal{F}a$ if possible such that its opposite face is also in the set. Then choose $\mathcal{F}b$ as one of the adjacent faces to $\mathcal{F}a$, giving priority in this choice to any adjacent face whose opposite face is in the set. If there are no opposite faces in the set, choose $\mathcal{F}a$, $\mathcal{F}b$ as any pair of adjacent faces.

This embraces the idea of giving priority to curves of shoulder points that cross over the interior of the cube, from face to opposite face. The pictures shown further give some insight about its suitability.

Examples of isosurfaces are shown in Figures 4.26 to 4.28. As a final note, one can argue that polygon $P^E{}_{3/0}$ is easy to triangulate so no bi-shoulder point is needed at all. Yet the example shown in Figure 4.12 shows how important a bi-shoulder point can be to improve the accuracy of the solution.



(a) $P^E{}_{3/0}$ (b) $P^E{}_{4/0}$



(c) $P^E{}_{5/0}$ (d) $P^E{}_{6/0}$

Figure 4.25: Triangulation of polygons P^E that do not lie on loop-back faces. We use one (magenta) bi-shoulder point for polygons that lie on (a) three, (b) four, (c) five and (d) six faces.

Topological polygons
$P^E{}_{3/0}$ and $P^E{}_{6/0}$

MC configuration: 12.ac

xyz	F_{xyz}
000	-1.5
001	1.0
010	-1.0
011	2.0
100	1.0
101	-1.0
110	-2.0
111	0.5

Bi-shoulder points
(0.574, 0.281, 0.508)
(0.759, 0.241, 0.640)

Inflection point
none

Body saddle point	F
(0.582, 0.042, 0.543)	-0.110

Pictures.
Our representation is on top whereas the "exact" isosurface of F is in the middle. The locus of shoulder points are represented at the bottom (red for slices of x constant, green for y and blue for z).

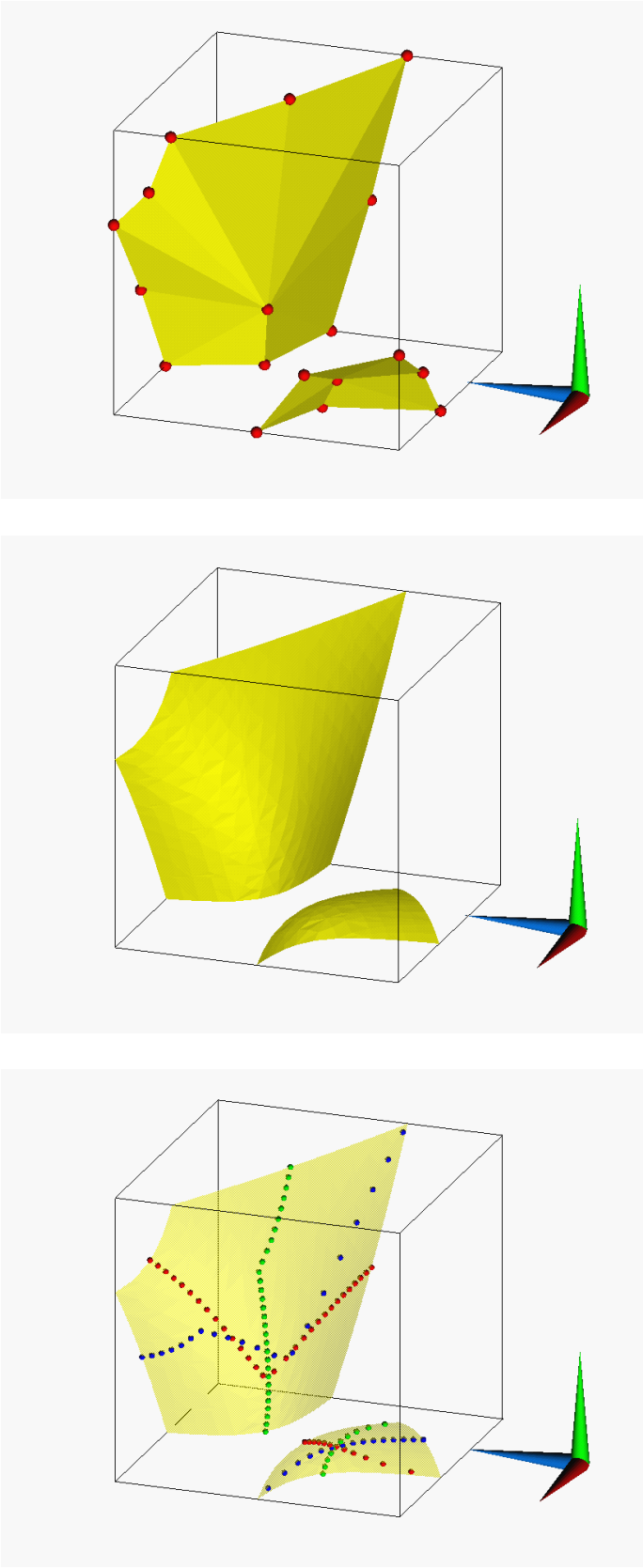


Figure 4.26: An example of $P^E{}_{3/0}$ and $P^E{}_{6/0}$.

Topological polygon
$P^E{}_{4/0}$

MC configuration: 2

xyz	F_{xyz}
000	4.0
001	-2.0
010	5.0
011	-8.0
100	-2.0
101	-1.0
110	-8.0
111	-1.0

Bi-shoulder point
$(0.317, 0.424, 0.077)$

Inflection point
none

Body saddle point
none

Pictures.
Our representation is on top whereas the ”exact” isosurface of F is in the middle. The locus of shoulder points are represented at the bottom (red for slices of x constant, green for y and blue for z).

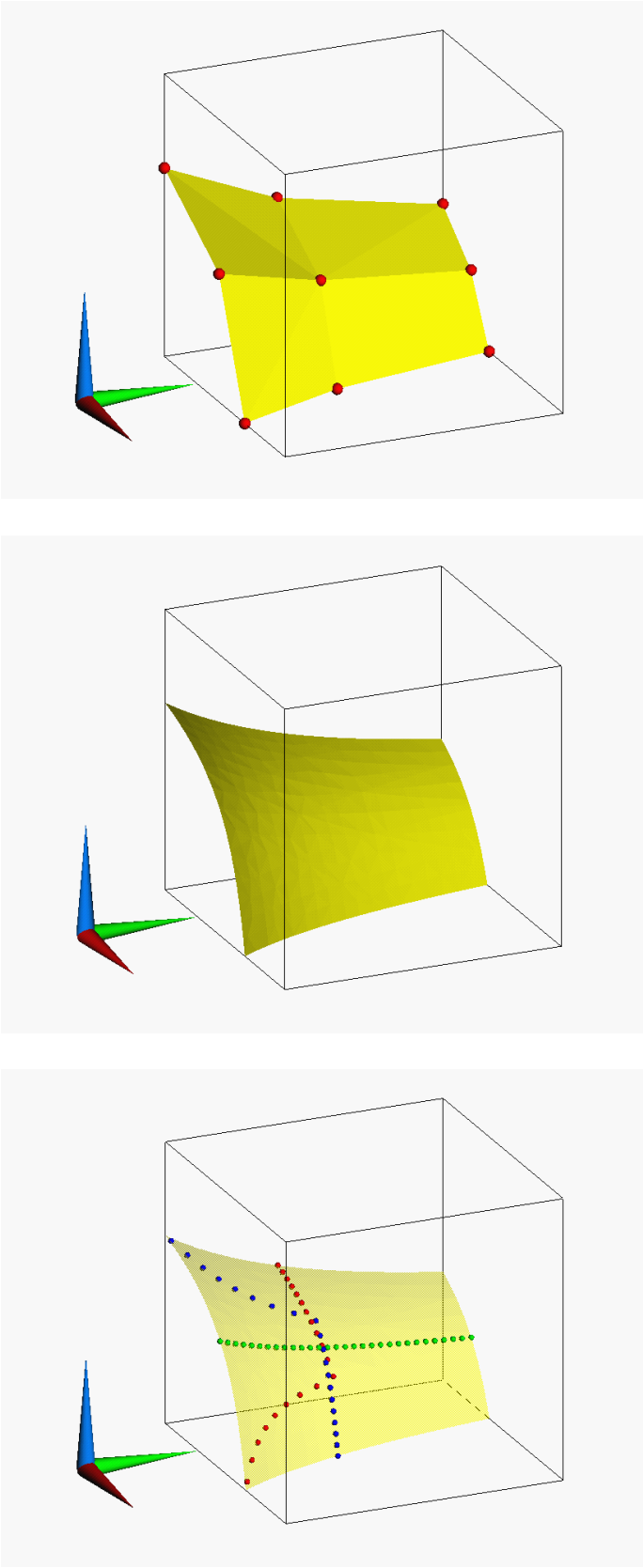


Figure 4.27: An example of $P^E{}_{4/0}$.

Topological polygon
$P^E{}_{5/0}$

MC configuration: 5

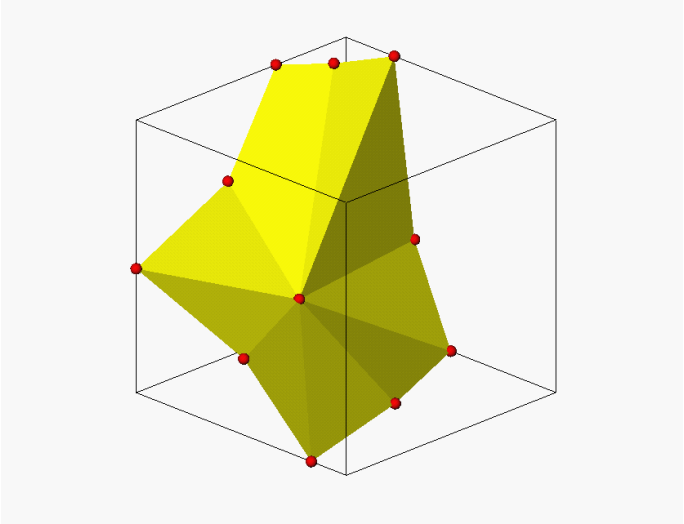
xyz	F_{xyz}
000	2.0
001	5.0
010	3.0
011	-6.0
100	-2.0
101	-1.0
110	-10.0
111	-4.0

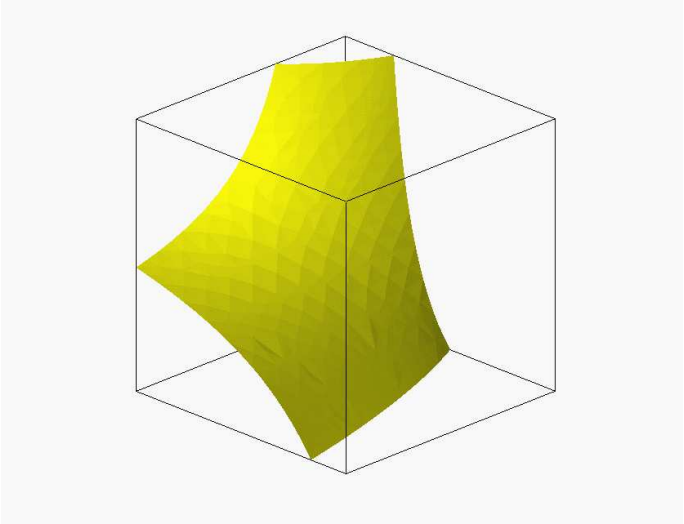
Bi-shoulder point
$(0.366, 0.328, 0.589)$

Inflection point
none

Body saddle point
none

Pictures.
Our representation is on top whereas the ”exact” isosurface of F is in the middle. The locus of shoulder points are represented at the bottom (red for slices of x constant, green for y and blue for z).





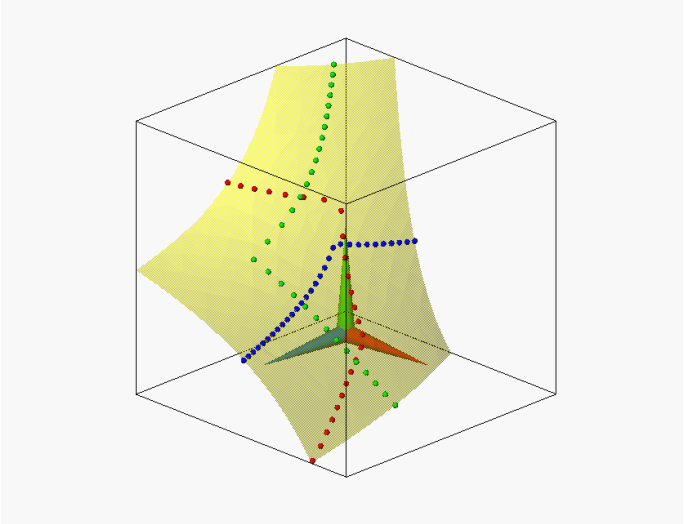


Figure 4.28: An example of $P^E{}_{5/0}$.

4.2.2.3 Tunnels

We now turn to the problem of internal tunnels. If one exists it is no more than a consequence of the interpolant F itself. First of all, how do we detect them?

Back to contouring, the asymptotic decider sets the topology in a cell face. It does so by comparing data values at cell vertices and the interpolant value at the point where its two first derivatives vanish, the so-called saddle point. As mentioned earlier in Section 4.1.1, Natarajan[76] extended the idea to 3D in order to detect tunnels. He introduced the concept of body saddle point: a point such that all three derivatives F_x, F_y and F_z vanish. Its formulae is:

$$x = -\frac{a + b\,z}{g + h\,z}\,, \qquad y = -\frac{(c\,f - b\,g) + (d\,f - b\,h)\,z}{d\,g - c\,h}\,, \qquad z = \gamma\,, \tag{4.37}$$

where γ satisfies the second-order equation

$$h\,(d\,f - b\,h)\,\gamma^2 + 2\,g\,(d\,f - b\,h)\,\gamma + (d\,e\,g - e\,c\,h + c\,f\,g - b\,g^2) = 0\,.$$

There are up to two solutions but only within the cube is of interest. So the method relies on the existence of the body saddle point and its corresponding F value. Let us explain how it works.

Suppose an isosurface $F = 0$ is represented by two pieces as shown in Figure 4.29(a). Suppose also in the volume in between those pieces the value of the interpolant F is greater than zero. The value in the remaining volume (excluding the isosurface itself) is negative. Yet if we change slightly the data at cell vertices, and so F, we can form a tunnel as shown in Figure 4.29(b). If that happens the values are now positive inside the tunnel whereas outside the values are negative. The body saddle point is internal to the tunnel and so its value is negative as well. This information about interpolant values allows us to detect the tunnel.

Now, consider the precise turning point of the transition from one situation to the other one. The body saddle point ought to be a point on the isosurface; the two pieces of surface touch each other. Actually this has similarities in contouring. There, the contour

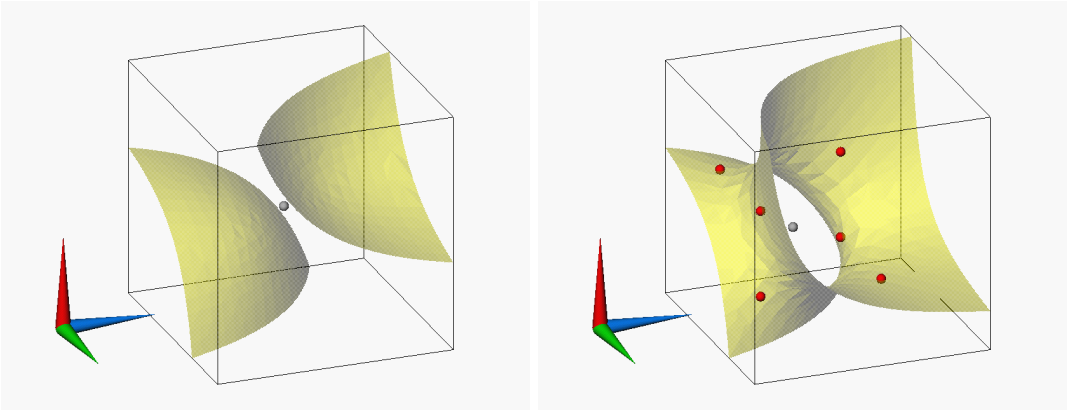


Figure 4.29: Formation of a tunnel. Moving from (a) two pieces of surface to (b) one surface with tunnel. The grey sphere is the body saddle point whereas the red spheres the six inflection points. Notice the relative positioning of the inflection points in accordance to Proposition 4.2.1.4 on page 83. This data example is the same as of Figure 4.6. The difference from (a) to (b) in terms of data is f_{111} whose value in (a) is 5.0 and in (b) 15.0 .

is degenerate, as two lines crossing at the saddle point. Returning to 3D, since $F = F_x = F_y = F_z = 0$, the body saddle point is also an inflection point. Moreover, an inflection point of any type: $\mathcal{I}x$, $\mathcal{I}y$ and $\mathcal{I}z$. Indeed we recall Proposition 4.2.1.5 on page 84 that states that all the second-order equations associated with the resolution of inflection points have the same discriminant. So when it vanishes, it vanishes for every type of inflection point. That is, if there is a solution it will be unique.

When the tunnel is formed the six inflection points exist. Now, there are two solutions for each type of inflection point rather than one as previously. From Proposition 4.2.1.4 on page 83, we know that if six inflection points exist then their location defines a cuboid and it is possible to draw a polyline linking them along the edges. Imagine the six inflection points and the body saddle point as geometric intersections of surfaces of type $F_x = 0$, $F_y = 0$ and $F_z = 0$ (second order equations). We conclude the body saddle point is to be within that cuboid. Notice however that, the existence of six inflection points does not necessarily imply the existence of the body saddle point. The counter example is found in configuration 13.i shown in Figure 4.24, where the polygon intersects all the edges of the cube.

On that basis, we propose the following:

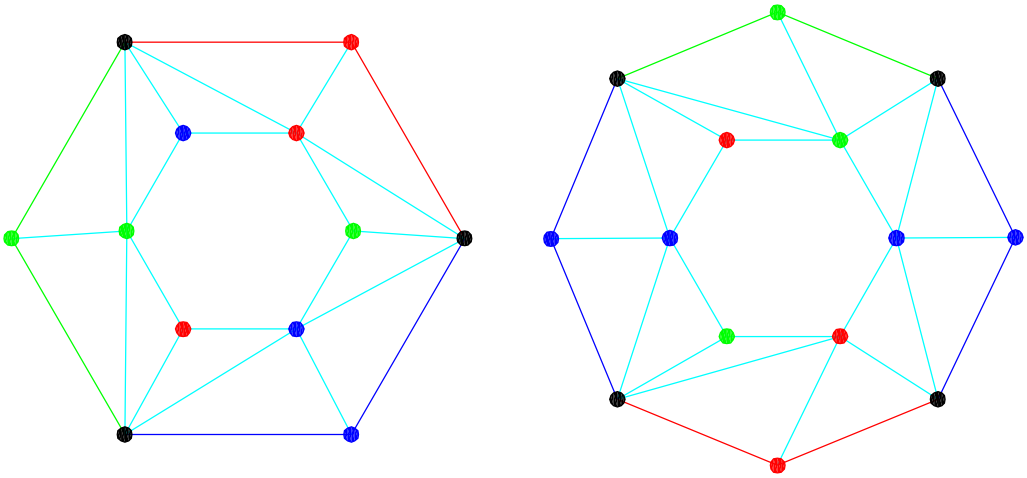
First, the detection of tunnels is based on the existence of two topological polygons — primary condition to be satisfied — and six inflection points. The reason why we do not use the method proposed by Natarajan is mainly due to implementation details. The three second order equations to obtain inflection points (that have same discriminant) are solved initially regardless of the detection of tunnels. Of course the number of topological polygons is known. On the other hand, the method of Natarajan sets a sign for each of the two topological polygons. This requires care when implemented; the spatial relationship is crucial since for each topological polygon there is one side positive, the other one negative.

Second, we use the polyline that was mentioned above, linking the inflection points, as the ”border” between the two topological polygons. Triangulation of each topological polygon is carried out independently. The important point is that they will join on that polyline and therefore there will be just one piece of surface.

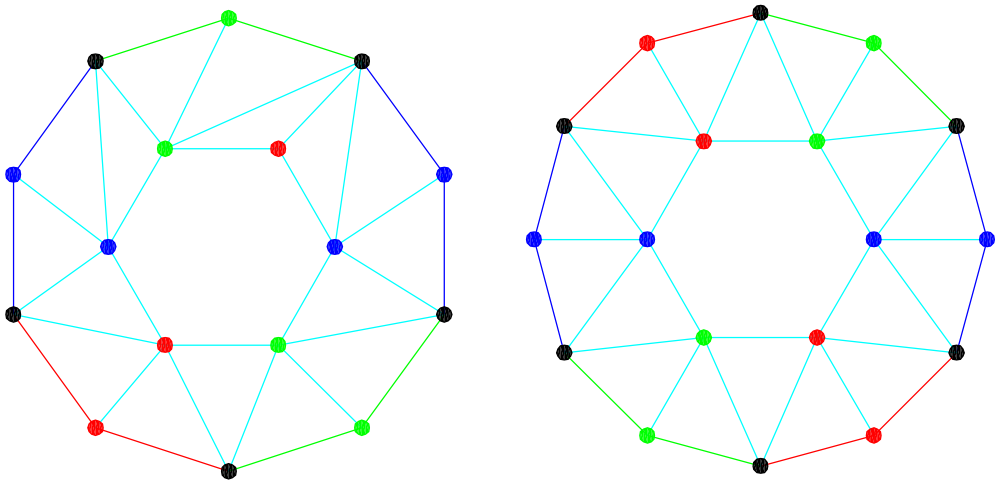
All the cases that can lead to tunnels are described on Table 4.4. It is interesting to note that those polygons do not lie on loop-back faces. Figure 4.30 shows the triangulation proposed for the polygons and then examples are provided, from Figure 4.31 to Figure 4.34.

Pairs of Polygons P^E	MC configurations
${P^E}_{3/0} \longleftrightarrow {P^E}_{3/0}$	4
${P^E}_{3/0} \longleftrightarrow {P^E}_{4/0}$	6.a
${P^E}_{3/0} \longleftrightarrow {P^E}_{5/0}$	12.ac
${P^E}_{3/0} \longleftrightarrow {P^E}_{6/0}$	7.d
${P^E}_{4/0} \longleftrightarrow {P^E}_{4/0}$	10.ac

Table 4.4: Pairs of topological polygons that can merge to create tunnels, and the underlying MC configurations (see also Table 4.1). The notation is according to Definition 4.2.2.1 on page 91 and Figure 4.30 shows the triangulation proposed.



(a) $P^E{}_{3/0}$ (b) $P^E{}_{4/0}$



(c) $P^E{}_{5/0}$ (d) $P^E{}_{6/0}$

Figure 4.30: Triangulation of polygons if involved in the formation of tunnels. They lie upon (a) three, (b) four, (c) five and (d) six faces.

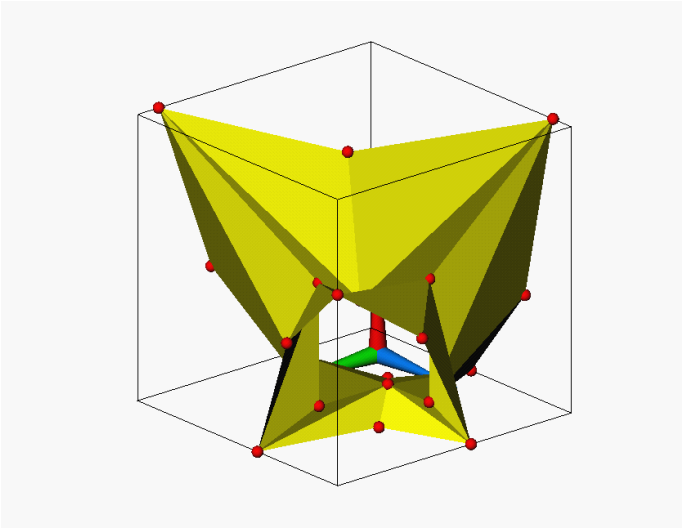
Topological polygons
$P^E{}_{3/0}$ and $P^E{}_{4/0}$ (tunnel)

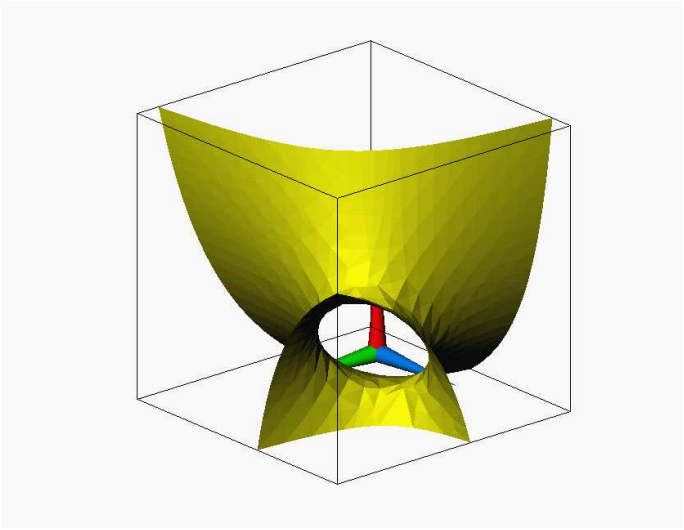
MC configuration: 6

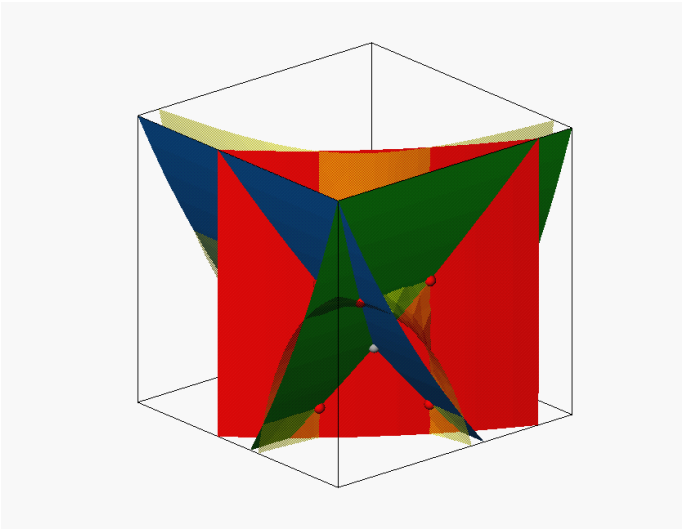
xyz	F_{xyz}
000	1.5
001	-1.5
010	-3.0
011	2.0
100	10.0
101	-1.0
110	-1.0
111	-1.0

Inflection points	
$\mathcal{I}z$	(0.106, 0.758, 0.624)
	(0.534, 0.462, 0.831)
$\mathcal{I}y$	(0.534, 0.758, 0.624)
	(0.106, 0.462, 0.831)
$\mathcal{I}x$	(0.536, 0.759, 0.831)
	(0.106, 0.462, 0.624)

Body saddle point	F
(0.313, 0.616, 0.731)	0.137







Pictures.
Our representation is on top whereas the ”exact” isosurface of F is in the middle. The surfaces in where the three first derivatives F_x (red) F_y (green) and F_z (blue) vanish are represented at the bottom, as well as the ”exact” surface (yellow but transparent), the body saddle point (grey) and the inflection points (red).

Figure 4.31: An example of one $P^E{}_{3/0}$ and one $P^E{}_{4/0}$ forming a tunnel.

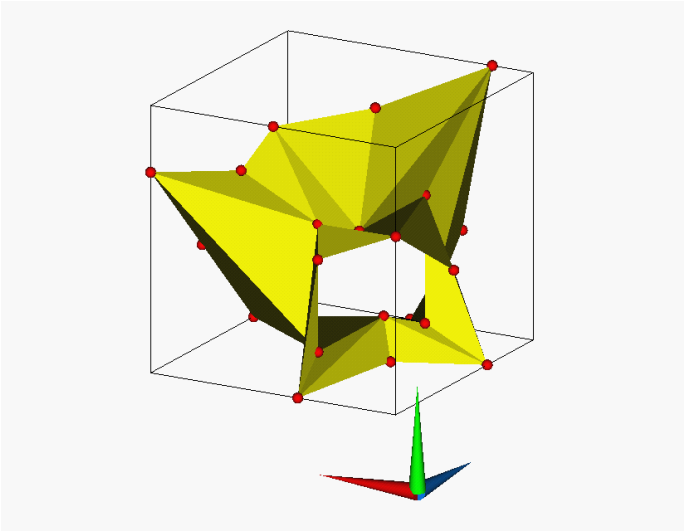
Topological polygons
$P^E{}_{3/0}$ and $P^E{}_{5/0}$ (tunnel)

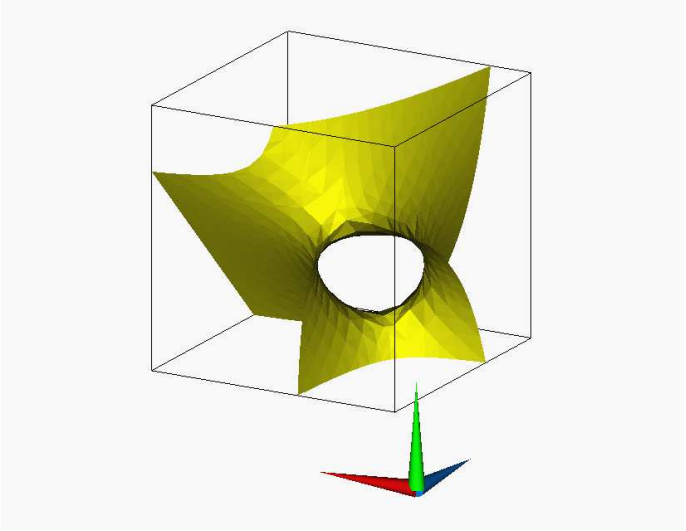
MC configuration: 12

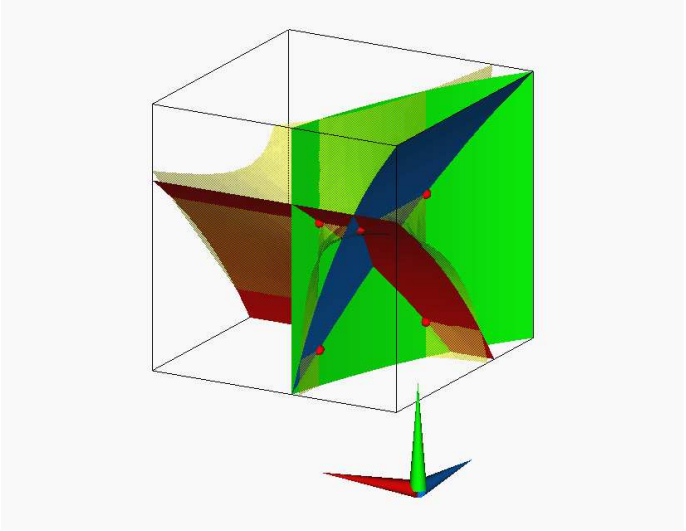
xyz	F_{xyz}
000	2.0
001	-1.0
010	-1.0
011	-1.0
100	-3.0
101	1.0
110	1.0
111	5.0

Inflection points	
$\mathcal{I}z$	$(0.393, 0.614, 0.136)$
	$(0.224, 0.136, 0.614)$
$\mathcal{I}y$	$(0.224, 0.614, 0.136)$
	$(0.393, 0.136, 0.614)$
$\mathcal{I}x$	$(0.393, 0.136, 0.136)$
	$(0.224, 0.614, 0.614)$

Body saddle point	F
$(0.314, 0.389, 0.389)$	0.117







Pictures.
Our representation is on top whereas the ”exact” isosurface of F is in the middle. The surfaces in where the three first derivatives F_x (red) F_y (green) and F_z (blue) vanish are represented at the bottom, as well as the ”exact” surface (yellow but transparent), the body saddle point (grey) and the inflection points (red).

Figure 4.32: An example one $P^E{}_{3/0}$ and one $P^E{}_{5/0}$ forming a tunnel.

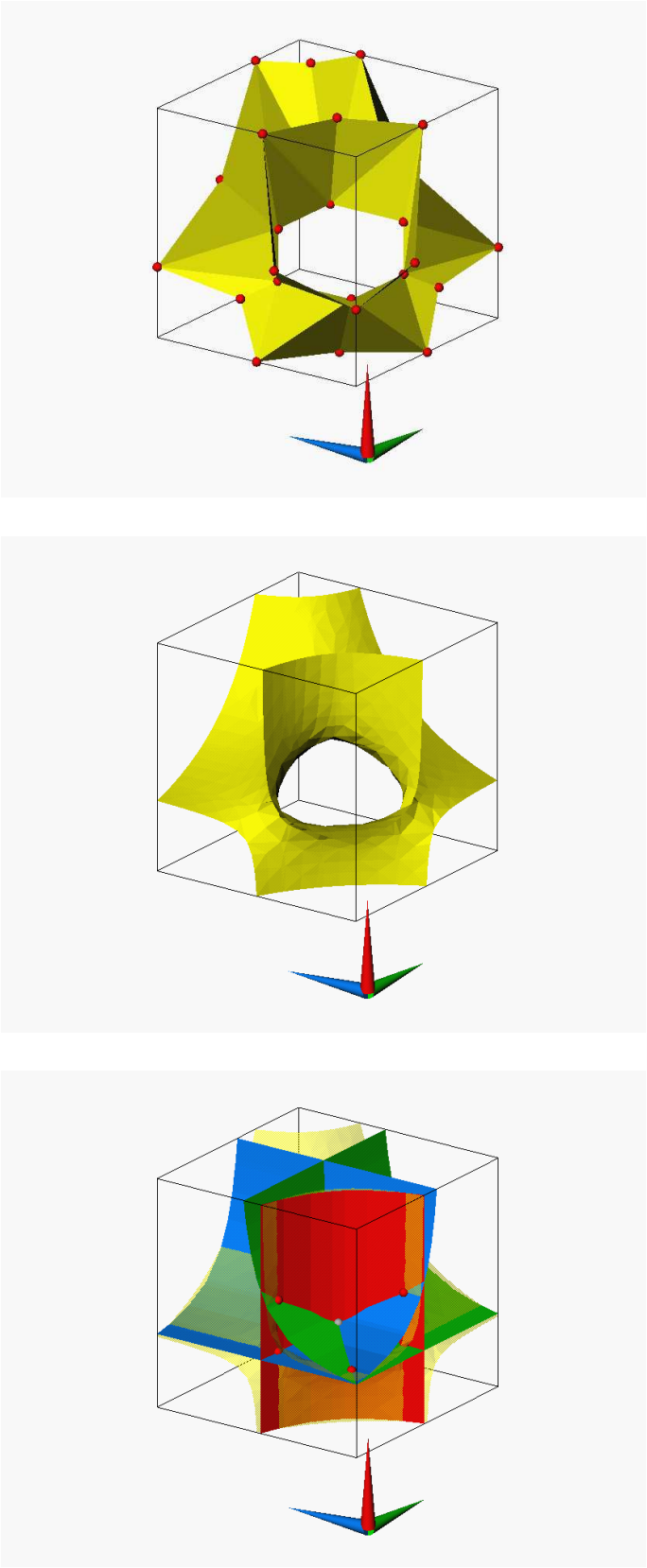
Topological polygons
$P^E{}_{3/0}$ and $P^E{}_{6/0}$ (tunnel)

MC configuration: 7

xyz	F_{xyz}
000	2.0
001	-2.0
010	-2.0
011	-2.0
100	-4.0
101	4.5
110	4.5
111	-2.0

Inflection points	
$\mathcal{I}z$	$(0.336, 0.454, 0.085)$
	$(0.563, 0.085, 0.454)$
$\mathcal{I}y$	$(0.563, 0.454, 0.085)$
	$(0.337, 0.085, 0.454)$
$\mathcal{I}x$	$(0.563, 0.454, 0.454)$
	$(0.337, 0.085, 0.085)$

Body saddle point	F
$(0.421, 0.316, 0.316)$	-0.127



Pictures.
Our representation is on top whereas the ”exact” isosurface of F is in the middle. The surfaces in where the three first derivatives F_x (red) F_y (green) and F_z (blue) vanish are represented at the bottom, as well as the ”exact” surface (yellow but transparent), the body saddle point (grey) and the inflection points (red).

Figure 4.33: An example of one $P^E{}_{3/0}$ and one $P^E{}_{6/0}$ forming a tunnel.

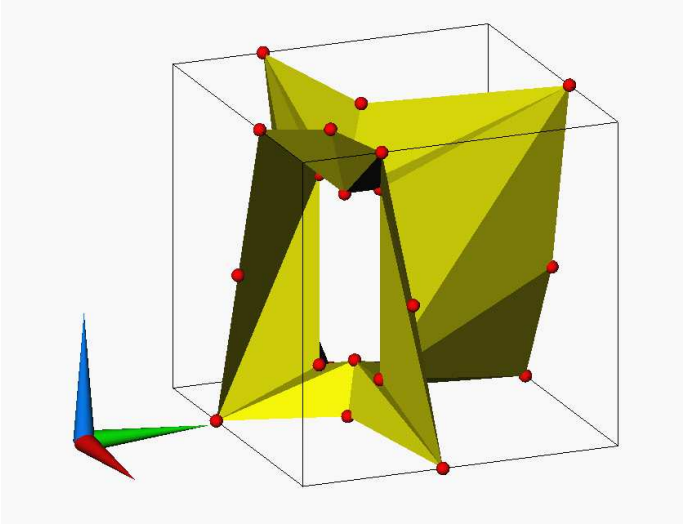
Topological polygons
$P^E{}_{4/0}$ and $P^E{}_{4/0}$ (tunnel)

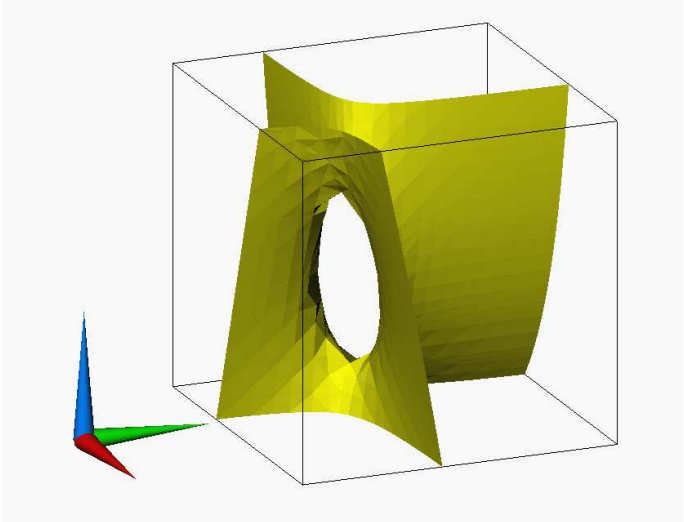
MC configuration: 10

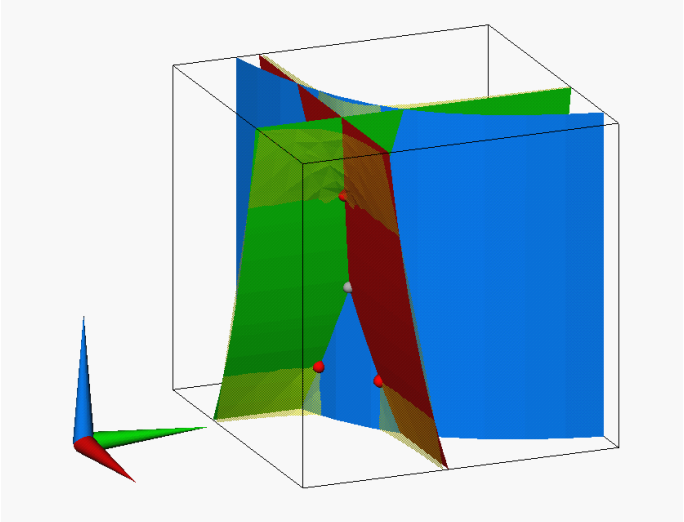
xyz	F_{xyz}
000	1.0
001	2.0
010	-1.0
011	-5.0
100	-2.0
101	-1.0
110	2.5
111	3.0

Inflection points	
$\mathcal{I}z$	$(0.383, 0.418, 0.150)$
	$(0.579, 0.305, 0.738)$
$\mathcal{I}y$	$(0.579, 0.418, 0.150)$
	$(0.383, 0.305, 0.738)$
$\mathcal{I}x$	$(0.579, 0.418, 0.738)$
	$(0.383, 0.305, 0.150)$

Body saddle point	F
$(0.489, 0.357, 0.421)$	-0.046







Pictures.
Our representation is on top whereas the ”exact” isosurface of F is in the middle. The surfaces in where the three first derivatives F_x (red) F_y (green) and F_z (blue) vanish are represented at the bottom, as well as the ”exact” surface (yellow but transparent), the body saddle point (grey) and the inflection points (red).

Figure 4.34: An example of two $P^E{}_{4/0}$ forming a tunnel.

4.3 Tests

We now present some images we have obtained using our method. Medical images are depicted in Figures 4.35 to 4.37 and in the Figure at the beginning of this Chapter, on page 62, whereas Figures 4.38 and 4.39 are based on Computational Fluid Dynamics (CFD) data. Tables 4.5 to 4.7 provide some statistical information as the nature of the datasets, the number of triangles generated and the frequency per type of the topological polygons P^E.

Dataset	Cells			Number of
	Total	Empty	No empty	triangles
Head	460,530	407,173 (88%)	53,357 (12%)	432,201
Brain	3,901,500	3,815,725 (98%)	85,775 (2%)	693,067
Spine	1,531,640	1,470,861 (96%)	60,779 (4%)	489,310
Pressure	2,601	1,583 (61%)	1,018 (39%)	8,144
Concentration SO_2	2,601	2,251 (87%)	350 (13%)	2,896

Table 4.5: Characteristics of the datasets and isosurfaces. It is shown information about the cells — the total number and how many are not (are) intersected by the isosurface — and the number of triangles generated. Not surprisingly, the majority of the cells are not intersected by isosurfaces.

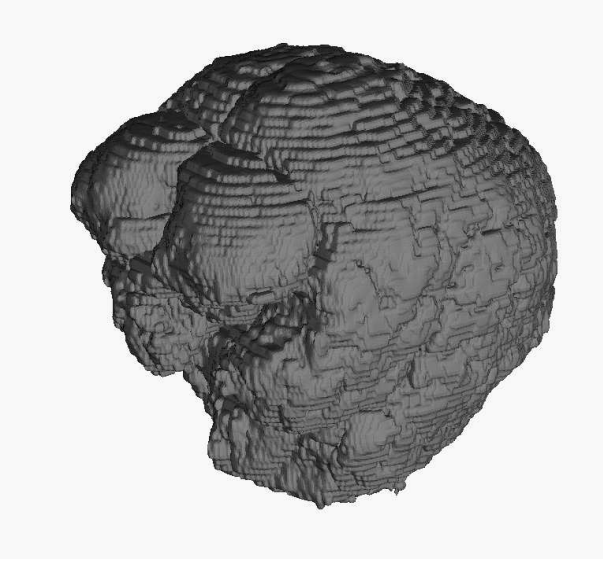


Figure 4.35: Brain. Details are presented in Tables 4.5 to 4.7.

Polygons P^E	Frequency				
	Head	Brain	Spine	Pressure	Concentration SO_2
${P^E}_{3/0}$	11,208	24,585	14,664	152	123
${P^E}_{4/0}$	33,893	43,175	33,264	729	153
${P^E}_{5/0}$	7,976	16,723	11,374	122	74
${P^E}_{6/0}$	663	2,073	1,578	15	12
${P^E}_{6/1}$	249	225	91	0	3
${P^E}_{7/1}$	136	323	77	0	1
${P^E}_{8/1}$	28	10	10	0	0
${P^E}_{8/2}$	25	19	6	0	0
${P^E}_{9/3}$	7	1	0	0	0
${P^E}_{12/3}$	0	0	0	0	0
${P^E}_{3/0} \longleftrightarrow {P^E}_{3/0}$	2	6	2	0	0
${P^E}_{3/0} \longleftrightarrow {P^E}_{4/0}$	2	3	2	0	0
${P^E}_{3/0} \longleftrightarrow {P^E}_{5/0}$	1	3	0	0	0
${P^E}_{3/0} \longleftrightarrow {P^E}_{6/0}$	0	0	0	0	0
${P^E}_{4/0} \longleftrightarrow {P^E}_{4/0}$	1	0	0	0	0

Table 4.6: Number of topological polygons P^E involved, for each test.

Polygons P^E	Frequency (% of total)				
	Head	Brain	Spine	Pressure	Concentration SO_2
${P^E}_{3/0}$	20.682	28.211	24.013	14.931	33.607
${P^E}_{4/0}$	62.544	49.543	54.470	71.611	41.803
${P^E}_{5/0}$	14.718	19.190	18.625	11.984	20.219
${P^E}_{6/0}$	1.223	2.379	2.584	1.473	3.279
${P^E}_{6/1}$	0.459	0.258	0.149		0.820
${P^E}_{7/1}$	0.251	0.371	0.126		0.273
${P^E}_{8/1}$	0.052	0.011	0.016		
${P^E}_{8/2}$	0.046	0.022	0.010		
${P^E}_{9/3}$	0.013	0.001			
${P^E}_{12/3}$					
${P^E}_{3/0} \longleftrightarrow {P^E}_{3/0}$	0.004	0.007	0.003		
${P^E}_{3/0} \longleftrightarrow {P^E}_{4/0}$	0.004	0.003	0.003		
${P^E}_{3/0} \longleftrightarrow {P^E}_{5/0}$	0.002	0.003			
${P^E}_{3/0} \longleftrightarrow {P^E}_{6/0}$					
${P^E}_{4/0} \longleftrightarrow {P^E}_{4/0}$	0.002				

Table 4.7: The same information as in Table 4.6 but in terms of percentage of the total number of topological polygons P^E for each test.

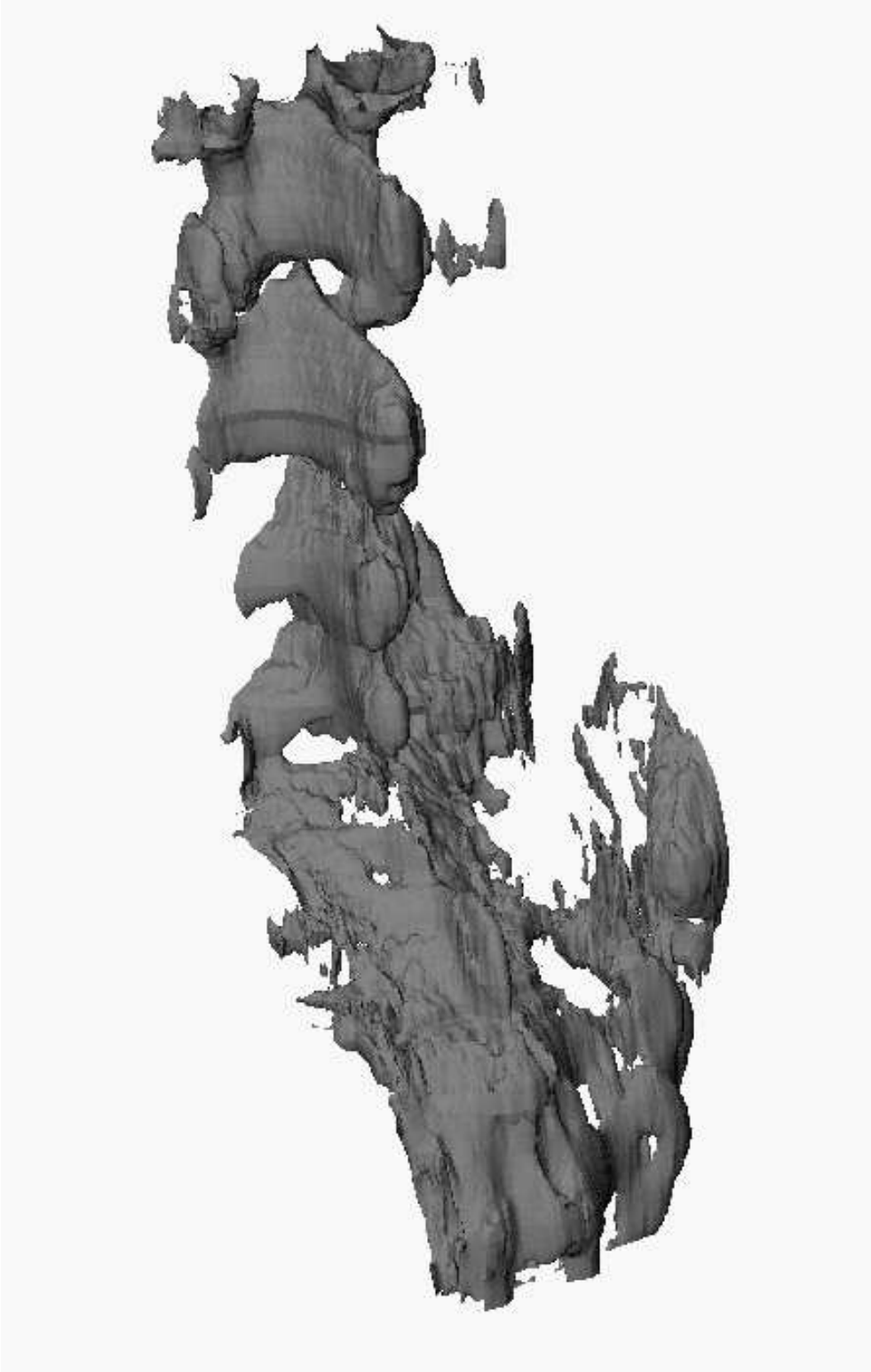


Figure 4.36: Spine. Tables 4.5 to 4.7 include related statistics.

Figure 4.37: Zooming of part of the spine shown in Figure 4.36.

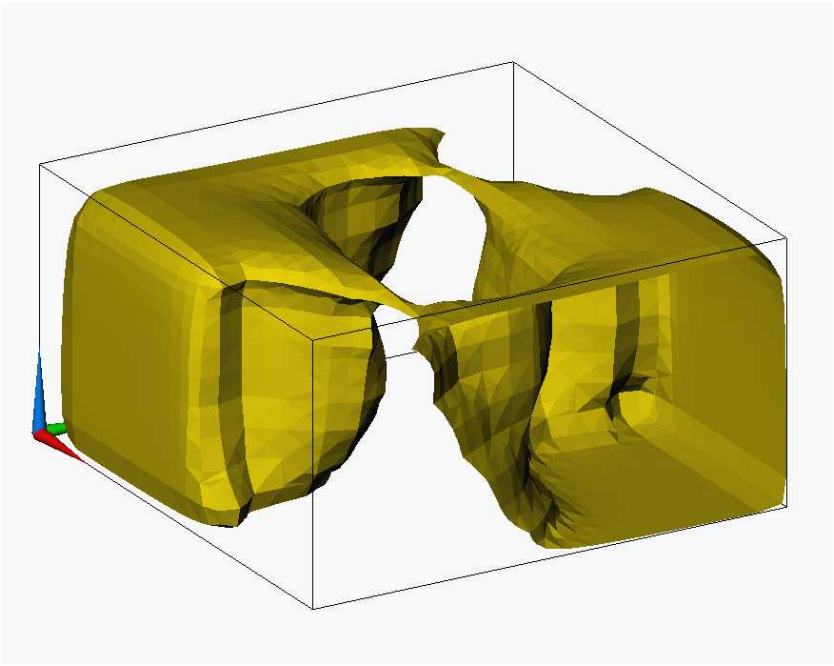


Figure 4.38: Pressure. Isosurface showing a pressure threshold of the air between two parallel sheets of glass — one cold, one warm — a simulation of the gap in double glazing. Statistics are included in Tables 4.5 to 4.7.

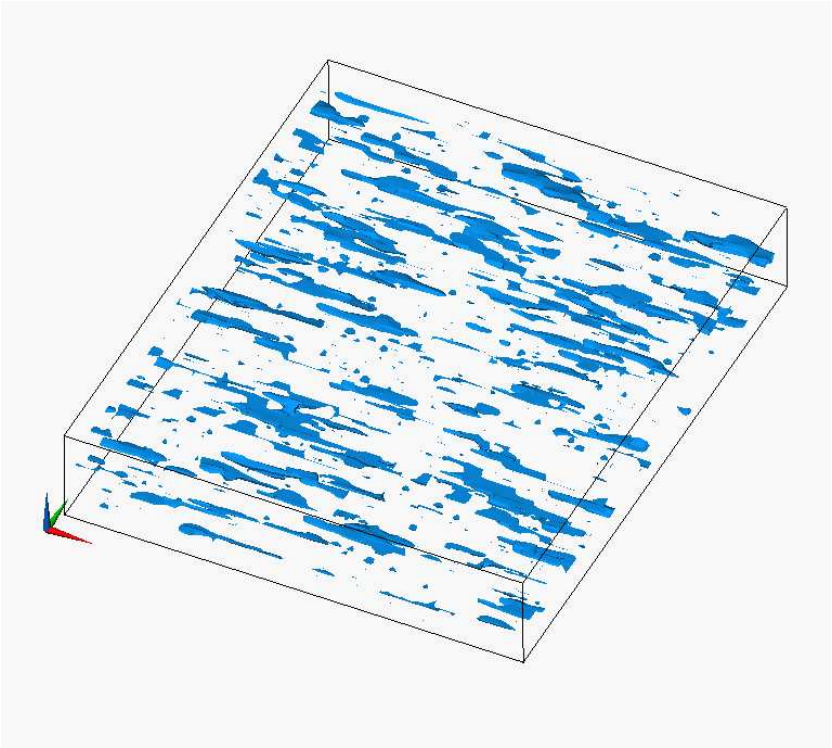


Figure 4.39: Concentration SO_2. Example of concentration of sulfur dioxide following an environmental model. Statistical information is shown in Tables 4.5 to 4.7.

4.4 Accuracy of Isosurfacing

In the previous Sections we have presented an isosurfacing method that we believe is very accurate. Nevertheless it is not free of uncertainty. Following the same steps as in contouring, we also would like to give an idea of the error committed. Unlike in contouring it is almost impossible here to obtain an exact metric for the error. In the following we present two approaches to depict accuracy.

In the first one, a error metric is approximated by using the area metric as defined in contouring (see 3.5.1 on page 55), in the faces of the cubic cell. Then the metric can be mapped by means of texture in the surface drawn. The metric may not be very reliable in this tridimensional space.

In a second approach, the analysis is made at the level of triangle drawn. Figure 4.40 helps to describe the idea. A result of our method is that the vertices of any triangle drawn lie on the ”exact” isosurface. So a closer approximation would be to use an extra point G on the ”exact” isosurface and then represent three triangles instead of just one (two vertices of the old triangle plus G). But at this stage we do not want to draw more triangles but simply to show how it could have been done. The visual paradigm to show uncertainty in this case can be three lines, from each vertex of the triangle converging to G. In all, rather than three triangles we will have the usual triangle plus three lines.

There is still the question of computing G. The goal is to have G such that the curvature of the isosurface in that neighbourhood is maximum. Or where F changes more rapidly. So we compute the gradient of the interpolant, ∇F, at the centroid of the triangle. Then from that point and with the direction of the gradient we compute the intersection with the ”exact” isosurface; G is obtained.

Finally we would like to mention that these ideas to depict accuracy are still under analysis. No implementation has yet been completed.

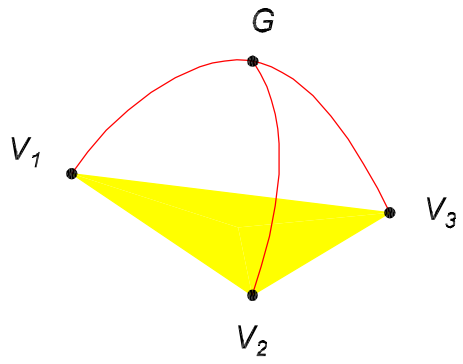


Figure 4.40: Use of an extra point G to depict a closer approximation to the isosurface. The three vertices V_1, V_2 and V_3 define the triangle drawn.

4.5 Conclusions and Future Work

We have carefully studied the Marching-Cubes method. Since its presentation, two major improvements were introduced: the use of the asymptotic decider to solve ambiguities in the faces of the cube, and the extension of this criterion into the interior of the cube.

Yet, we realised that more details were to be unfold. In respect to the configurations, not only did we find a new subcase, but we also proved mathematically that some of the subcases already presented were impossible to happen. We also concluded that, although the problem of tunnels in the interior of the cube was discussed in a correct way, assuming trilinearity within the cube, not enough attention was paid to the way triangulation was carried out. In particular, it should be observed that, since a surface has to be continuous and not self intersecting, no triangles are allowed to lie in the faces of the cube.

As a consequence of the work on contouring, and our understanding of the MC method, we developed a more accurate Marching-Cubes method. The aim was to represent as accurately as possible the trilinear interpolant within each cell. Positional continuity of the isosurface across adjacent cells was guaranteed as in the MC; correct topology in the faces of each cell was also verified, since the asymptotic decider is used.

This accurate Marching-Cubes method relies mainly on supplementary points on the isosurface, either lying in the faces, or in the interior of the cell. In the faces of the

cube we use face shoulder points as supplementary points, similarly to contouring. For the interior of the cube, our strategy was based on the observation that the isosurface can be seen as a pile of consecutive parallel contours. These supplementary points in the interior are inflection points — points such that two first derivatives of the trilinear interpolant vanish — or else bi-shoulder points — points that are simultaneously shoulder points on the contour of the trilinear interpolant on two orthogonal slices through the cube. An inflection point reflects the topology in the interior of the cell, whereas a bi-shoulder point is more related to aspects of accuracy, with no influence in the way topology is defined.

A major aspect we stressed out was the triangulation of the isosurface. In comparison to the classic MC, our method takes a different view: rather than following configurations, we are driven by topological polygons, no matter from which MC configuration they arise. There are three situations to consider: polygons lying on loop-back faces, polygons not lying on loop-back faces, and situations of tunnels in the interior of the cell. In the first case, the triangulation uses inflection points. In the second case, one bi-shoulder point and in the latter one, inflection points are used. We found interesting features in the formation of tunnels. First, a tunnel is formed upon two topological polygons not lying on loop-back faces. Second, there are six inflection points. In that respect, we noticed that if there are six inflection points then all are located in the corners of a cuboid. This result is the foundation for the triangulation of tunnels. A polyline is defined linking all the inflection points along the edges of the cuboid. On that basis, each polygon is triangulated independently, being the polyline mentioned as the ”border” between the two polygons.

We are aware that an alternative way to represent the trilinear interpolant to greater accuracy is to pursue a crude but simple approach — namely, to subdivide each cube into many smaller cubes, and applying the MC algorithm within each minor cube (indeed the pictures in the thesis depicting ”exact” isosurfaces were drawn that way). However the aim of our approach has been to use a more ”intelligent” approach were we try to represent the fundamental topology of the trilinear interpolant within the original cube, with a minimum of additional points. This has the significant advantage of course in practice of not generating an inordinate number of triangles.

As far as further developments are concerned, we should point out the following:

First, triangulation of polygons that do not lie on loop-back faces, so involving bi-shoulder points, deserves more study. The bi-shoulder points are computed numerically which means more computational time. In addition, more than one bi-shoulder point can be defined in the majority of these polygons, but so far we are just using one. Maybe we should use more, and therefore the solution will be more accurate, or else we can define extra guidelines to choose the most appropriate bi-shoulder point, apart from the rule of thumb we have set out.

Second, close attention has to be paid to the number of triangles generated. That is, to consider decimation of triangles, as long as users are willing to accept smaller accuracy, and in which circumstances. For instance, to establish guidelines as follows: in case of triangulation of topological polygons that do not lie on loop-back faces, two adjacent triangles that have a face shoulder point as common vertex should merge if the angle due to that shoulder point implies nearly flat triangles.

Third, to combine the new method with mechanisms that avoid the waste of time on processing cells that do not contribute to the isosurface. Techniques to do so can be found in the references 1, 17, 51, 63, 97, 120. They usually require information to be held in memory throughout.

Fourth, the representation of interval volumes. That is, rather than showing an isosurface $F(x,\, y,\, z) = \Theta$ as we do now, the objective is to show an interval volume such as $\alpha \leq F(x,\, y,\, z) \leq \beta$.

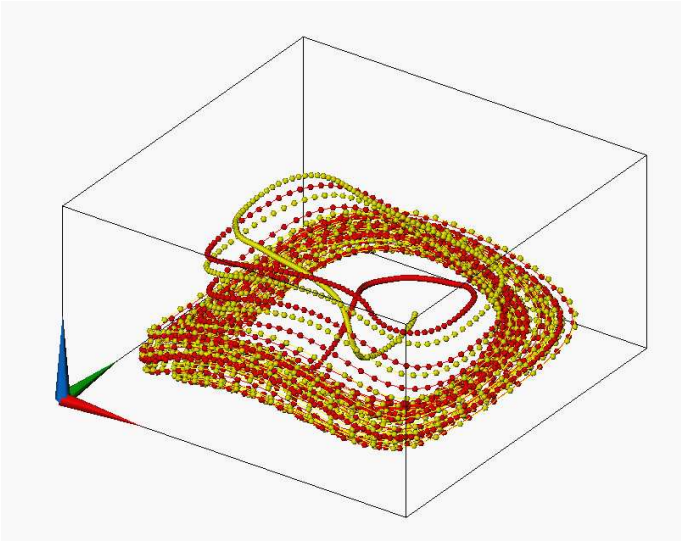
Fifth, to implement the ideas we have presented concerning the representation of accuracy. Recall that two suggestions were made: in one of them, and similarly to what we have done in contouring, an error metric is mapped using texture on the isosurface. The other idea involves the computation of an extra point lying on the surface, close to each triangle drawn. Then, to show accuracy, lines are drawn between this new point and the vertices of the triangle.

It is noteworthy that the last suggestion may eventually lead to further improvements: the replacement of the triangle by three new ones. The process can even be recursive, of course as long as it is affordable to do so.

Chapter

FIVE

Particle Tracing



Particle tracing is a simple but effective technique used in flow visualization, of which the figure above is an example. It is a fundamental technique which forms the basis of a variety of techniques, such as streamribbons and streamsurfaces.

In this chapter we start by introducing the technique. Particular attention is devoted to issues such as data spaces, integration schemes, velocity interpolation and the selection of the step size. Next we address the topic of visualizing the accuracy of particle tracing. There are three techniques to mention: re-integration, global error estimators

and velocity residuals. Each one of these is discussed and implemented. Alongside that we introduce visual paradigms to convey the accuracy data before we present some trace of particles. The data set used to illustrate the work is a representation of the air flow between two parallel sheets of glass. Finally we comment on the overall results and point out further developments.

5.1 The Technique

In the context of computational fluid dynamics, the aim of particle tracing is to track the motion of a particle released at some point through a flow field. The field is discrete, both in space and time. Let us limit the discussion to massless particles, that is, to particles following the local flow field.

Thus the problem is posed as follows: given the velocity field $\vec{V}(p,\, t)$ and the position p_t of a massless particle at an initial time $t = 0$, within the flow field, calculate the successive particle positions. The motion is ruled by the Ordinary Differential Equation (ODE) with initial value:

$$\frac{d\,p}{d\,t} = \vec{V}(p,\, t)\,, \quad \text{given } p_0\ . \tag{5.1}$$

Most of the methods rely on numerical integration schemes. Hence the motion from one position p_i to the next one p_{i+1}, after time Δt, is determined by:

$$p_{i+1} = p_i + \int_{t_i}^{t_{i+1}} \vec{V}(p_t,\, t)\ d\,t\ . \tag{5.2}$$

A generic particle tracing method comprises the following steps: first, the grid cell where the particle lies at the initial time is identified. Second, the velocity at that location is computed. In order to do this, some sort of interpolation is required since the velocity field is discrete. Next, the particle is repositioned using a numerical integration that evaluates (5.2). The whole process is repeated until the particle leaves the domain or its velocity vanishes. Furthermore the computation may stop after a given period of time.

An important issue in particle tracing is the errors made at each step. They tend to propagate as the computation proceeds. We can name at least three sources of errors. They are:

– the numerical integration;

– the velocity interpolation;

– the velocity data itself.

Henceforward a local error is defined as the error made at a single time step while a global error measures the cumulative effects of the local errors from every time step, including those from startup procedures inherent to the integration scheme.

The suitability of any method is very much dependent on the nature of the velocity field and on the accuracy desired. Particle tracing is much easier in steady (laminar) flows — the time scale of the flow is so negligible that the velocity is considered time independent — than in unsteady (turbulent) flows. The Jacobian matrix $J = \partial \vec{V} / \partial t$ is crucial as far as accuracy is concerned. The way the grid domain is discretised is also an important aspect. In the following we will discuss some of these issues.

5.1.1 Physical Versus Computational Space

The flow field is usually defined over a curvilinear grid — the so-called physical space. However to facilitate computations, the curvilinear grid is sometimes mapped into a Cartesian grid — the computational space. One main reason to do so is that, during particle tracing, there is a need to locate the grid cell in which the particle lies. This cell search is not an easy task to perform in curvilinear grids. But it is rather simple in the case of a Cartesian grid.

Unfortunately there are some problems with the conversion of spaces. First, there is no global transformation but only a local transformation. This depends on the local Jacobian, the matrix of partial derivatives. In the end the conversion between spaces is very costly. Second, the accuracy of the physical space may be lost. For example it is

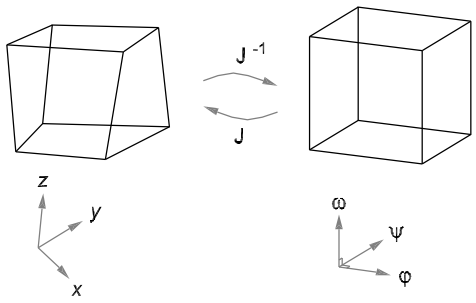


Figure 5.1: Transformation between a physical (curvilinear) space and a computational (Cartesian) space. It requires a local transformation for each cell based upon the partial derivatives, *i.e.* the Jacobian J.

possible to obtain very large velocities when rather deformed grids are mapped into a Cartesian grid.

Furthermore, we would like to remark that there is substantial improvement in the cell search operation if the physical space is decomposed into tetrahedra.[57]

5.1.2 Integration Schemes

There are a range of methods for the solution of ODEs such as (5.1). The common ones in particle tracing are the Euler method, and the multi-stage methods such as the explicit Runge-Kutta (RK) methods of second and of fourth order. Other methods which can be used are for example the multi-step schemes such as Adams-Bashforth, Adams-Moulton and backwards differentiation.

The choice of one of these integration schemes is mainly driven by the error order. The Euler method (first order) is widely assumed to be poor whereas the higher order RK methods are acceptable. Nevertheless the errors due to other sources such as velocity interpolation can be a decisive factor. For instance a flow data has been generated as second order accurate in time, then the integration scheme should be at least of error order three. Then the error due to integration is less then the error due to the simulation.[23]

Another point worth mentioning is the complexity to initialise higher-order methods. It usually requires other methods to provide the first samples. In this way these startup errors will contribute to define the accuracy of the overall technique.

Methodology	Formulae	Local error
Euler	$p_{i+1} = p_i + \Delta t\ \vec{V}(p_i)$	$O(\Delta t^2)$
2nd-order Runge-Kutta	$p_{i+1} = p_i + \frac{\Delta t}{2}\ \vec{V}(p_i + p^*_{i+1})$	$O(\Delta t^3)$
4th-order Runge-Kutta	$a = \Delta t\ \vec{V}(p_i)$	$O(\Delta t^5)$
	$b = \Delta t\ \vec{V}(p_i + (a/2)\,)$	
	$c = \Delta t\ \vec{V}(p_i + (b/2)\,)$	
	$d = \Delta t\ \vec{V}(p_i + (c/2)\,)$	
	$p_{i+1} = p_i + \frac{1}{6}\,(\,a + 2\,b + 2\,c + d\,)$	

Table 5.1: Multi-stage integration schemes for steady flows. We present the Euler and the predictor-corrector RK of second (also known as midpoint or Heun method) and fourth order. $\vec{V}(p_i)$ is the instantaneous velocity at position p_i, and $\Delta t = t_{i+1} - t_i$ is the incremental time step. In case of 2nd-order RK the p^*_{i+1} is computed using the Euler method.

5.1.3 Velocity Interpolation

One of the tasks to be carried out during particle tracing is to compute the velocity at some point within a grid cell. A common choice is to use trilinear interpolation based upon the known velocity values at the cell vertices. Bear in mind that the velocity field is discrete. However this interpolant may not convey accurately the field. For example it provides poor results when there are sharp gradients in the field. In that case the errors due to interpolation may be higher than the ones of the integration.[13] Therefore higher-order interpolation schemes are a valid option as well. Unfortunately the additional computational cost involved with higher-order interpolants can be so huge that it makes its usage rather difficult.

Another situation to be considered is that of interpolation in time. This is the case with unsteady flows, that vary not only in space but in time. So if a required time is not available from the grid, then a temporal interpolation of velocity is performed before the spatial interpolation occurs.

5.1.4 Step Size

The step size is crucial in defining the next particle position. Theoretically it can be fixed or adaptive. But in general the choice is to have an adaptive step size. If the step size is too large, then the particle may miss important characteristics of the flow such as points where the velocity changes dramatically. It may be inaccurate in this case. But on the other hand, if the step size is too small the computation may be very costly and without substantial compensation in terms of accuracy. The general aim is to maintain the accuracy within the cell and avoid excessive steps when the velocity is reduced. One guideline is to have an adaptive step size proportional to the size of the cell and to the inverse of the magnitude of velocity.[13] Another one is to use the curvature of the particle trace, which in the end relates to the gradient of the field; the higher the curvature the lower the step size should be.

5.2 Visualization of Accuracy

Our main goal in this Chapter is to show accuracy in particle tracing. We believe users should understand the degree of error committed by a particular particle tracing technique, before engaging themselves to any judgment.

Before going further we would like to refer to some work by other authors. The majority of examples concentrate on comparison of different methods and normally use analytical models to assess the accuracy of these different methods. This is the case of Darmofal and Haimes[23] who present a valuable numerical analysis of accuracy and stability of various particle tracing algorithms, particularly for unsteady flows. Lodha *et al.* [64] focused on visualizing the differences in particle traces obtained from various integration schemes and/or different step sizes. Walton[116] shows results from two different integration methods. Knight and Mallison[61] compare their own methodology based on stream functions with common particle tracing methods.

5.2.1 Strategies

The primary issue is to establish how to assess the accuracy of a given particle trace. If the exact solution was known in advance then the difference between both would be the answer. That can only happen if the data sets are analytical. But in that case, the analysis is more academic rather than practical.

Let us concentrate on errors due to integration. Hence, the goal is to obtain a quantitative measurement of the integration error. On the basis of common error control strategies used in the solution of ODEs we present three approaches to derive accuracy measurement for particle tracing. They are:

Re-integration.

A second but different integration is carried out. For example, with smaller tolerance. The discrepancy between both integrations gives a measure of accuracy.

Global error estimators.

As integration proceeds the magnitude of the global error at each integration point is estimated.

Velocity residuals.

As integration proceeds a residual of velocity is computed at sampling times. That is, the difference between the first derivative inferred by the numerical solver and the velocity obtained from the flow field. It is an indirect measurement. The bigger the residual the lower the accuracy.

5.2.2 Framework

Given the strategies presented above, we can now establish the practical framework. It is our wish to follow a method widely used, in particular a particle tracer from a commercial visualization system. So, in this study the framework is as follows:

1. Use of a RK scheme with an adaptive time-stepping, as in the IRIS Explorer module NAGAdvectSimple.

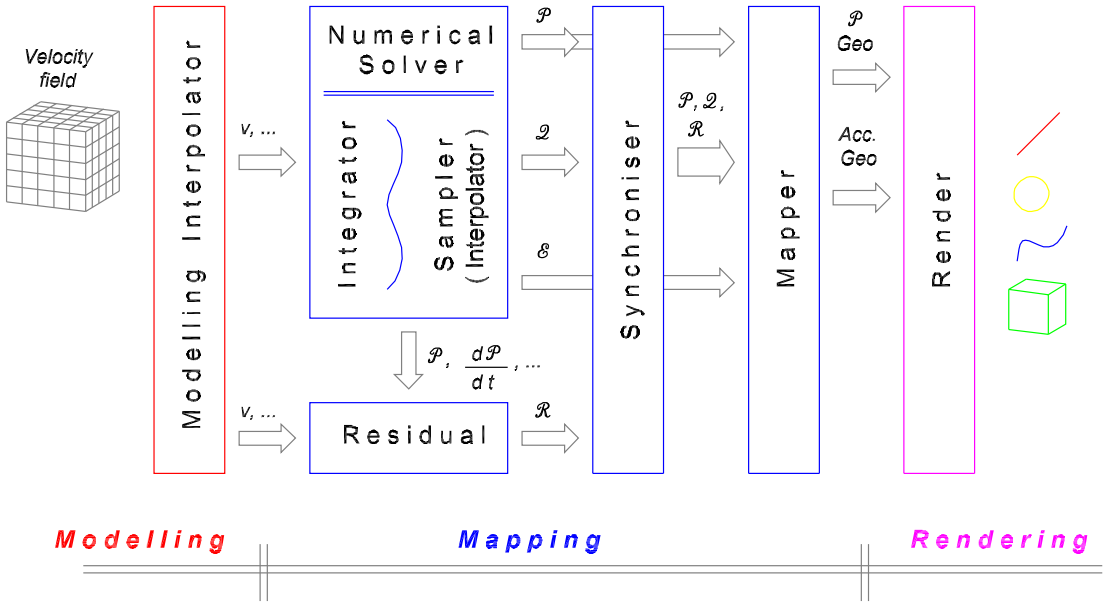


Figure 5.2: Architecture to visualise errors due the numerical solver in particle tracing. The core is the numerical solver. $\mathcal{P}$ is the main trace, $\mathcal{Q}$ is a second trace by re-integration, $\mathcal{E}$ the global error estimators and $\mathcal{R}$ the velocity residuals. Notice that this architecture embraces the data flow model of scientific visualization, from modelling to rendering.

2. To exclusively concentrate at this stage on errors due to the RK scheme used.

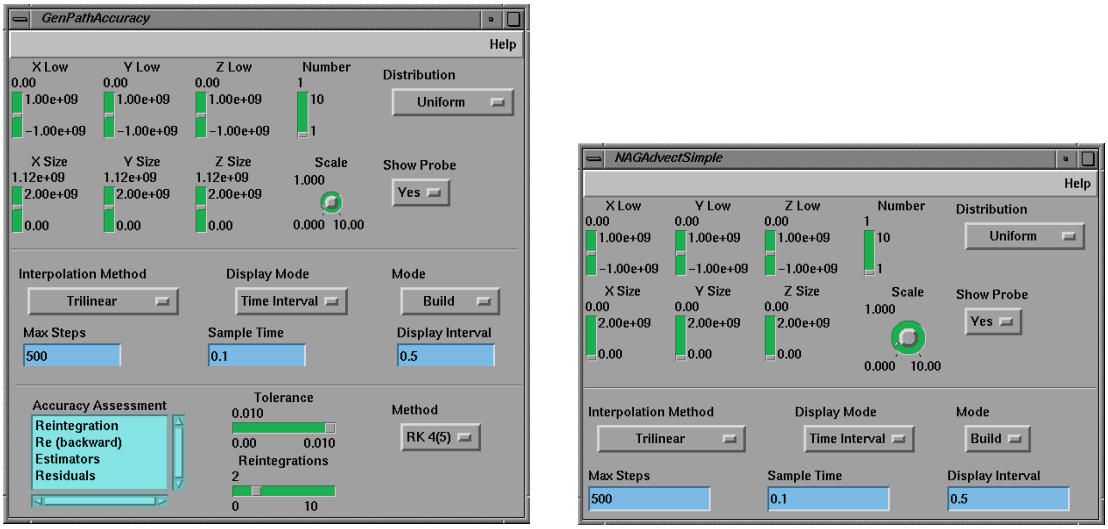
3. Show particle positions according to a specific sampling frequency.

4. Use of a practical rather than a theoretical flow data set.

This is the framework within which we shall explore our ideas to visualize accuracy in particle tracing.

5.2.3 Architecture

To deliver the analysis we are looking for we propose the architecture shown in Figure 5.2. The main component is the numerical solver. It delivers the main trace $\mathcal{P}$ (the one against which comparisons are made) and if required a second path $\mathcal{Q}$ to be obtained by re-integration. It can also provide inferred velocities from integration in order to compute the velocity residuals $\mathcal{R}$. Finally the integrator within the numerical solver can provide the global error estimators $\mathcal{E}$. These strategies may or may not be followed simultaneously.



(a) (b)

Figure 5.3: Implementation of (a) GenPathAccuracy as an extension of (b) IRIS Explorer NAGAdvectSimple. GenPathAccuracy is based upon the architecture shown in Figure 5.2.

At the beginning of the pipeline there is the modelling interpolator. Its purpose is to obtain velocities from the flow field given the positions in the grid. This is the modelling stage. As mentioned before we do not take into consideration errors due to velocity interpolation, nor those due to the (simulation of) data for the time being.

Next there is the computation of the main trace and associated accuracy measurements. An important point at this stage is the necessary synchronisation of the data that is computed. Further down the pipeline this data is converted into geometries and then rendered. If for example there are two traces whose time steps do not coincide we must ensure anyway that visual comparisons are done at exactly the same time. This synchroniser unit behaves as a filter in time.

This architecture is implemented as two modules in a Open Inventor / IRIS Explorer environment. We have increased the functionality of the IRIS Explorer NAGAdvectSimple to accommodate accuracy measurements — GenPathAccuracy as shown in Figure 5.3 — and built a new module specifically to map this accuracy data into geometries.

5.2.4 Data Generation

The generation of data is the primary issue in the whole process of visualizing accuracy. We can only give a sensible view of the accuracy if we compute reliable error estimation. Next, we discuss all the major issues as far as generation of data is concerned.

Numerical Solver. As with NAGAdvectSimple, we use the routine nag_ode_ivp_rk_onestep for initial value problems in ODEs from the NAG C library. Two pairs of RK formulae can be used: one pair with formulae of error orders 2 and 3, and the other of 4 and 5. For each coordinate direction i, the result at each integration step is always from the higher-order formula (local extrapolation) and it is the difference between results within the pair that determines the local error estimator E_i^L.[6] To control the step size this value is measured relative to a weight W_i, defined as:

$$W_i = MAX \left(\frac{|\, p_b \,| + |\, p_e \,|}{2}, T_i \right) , \tag{5.3}$$

with p_b and p_e the solution at the beginning and end of the step respectively. T_i is the threshold value set as:

$$T_i = t * MIN \ (\, |\, L_i^a \,|, |\, L_i^z \,| \,) \ , \tag{5.4}$$

where t is the relative error tolerance set, and L_i^a, L_i^b the two data set boundaries in relation to coordinate direction i. Taking into consideration all the coordinate directions, the step size is controlled such that

$$MAX \left| \frac{E_i^L}{W_i} \right| \ < \ t \, . \tag{5.5}$$

Note that t and W_i are constrained. They must be such that:

$$10.0 * mp \leq t \leq 0.01 \,, \tag{5.6}$$

$$W_i \geq +\sqrt{sn} \,, \tag{5.7}$$

where mp and sn are respectively the machine precision and the smallest positive number representable on the machine.

Results of particle positions are usually required at specific sampling times — crucial in animation for example. So the numerical solver has interpolants associated with the RK formulae, with the same order of accuracy, to sample in time the results from integration. In the same way they can offer the first derivatives at those sampling times.

Re-integration. This technique looks at different solutions of the same problem. Integration is repeated but in a different framework. We set up three variants:

– Re-integration with smaller relative error tolerance, in this case reduced by a factor of 10.

– Re-integration in backwards direction. The last point in the first integration is the starting point for the second integration. The time direction is therefore reversed.

– Many re-integrations with different relative error tolerances, say reducing by 1/10, 1/100 and so on. These results can provide a valuable insight into the trade-off between accuracy and computational cost.

All the solutions consist of particle positions and related time scale.

Global error estimators. Global error assessment aims to show the cumulative effects of the local errors (that control the integration) at all the integration steps. In theory it constitutes the best way to assess accuracy. When required, the NAG solver can deliver the Root Mean Square (RMS) average of the error at any integration point of interest, taken from the beginning of the integration.

The method is as follows: for each integration a(more accurate) subsidiary integration is performed with two half steps over the same interval. The assessment is based on the difference of results between both integrations.[95]

However the theory behind such an approach is no longer valid when there is discontinuity in the derivative function, or even when the tolerances used are at the extremes of the range.[96] As a consequence, if at some point the assessment is not credible (for example if the estimated local error in the subsidiary integration is not sufficiently inferior to that of the main integration) then any assessment onwards is discarded as well.

We should point out that reported solutions are different depending on whether global error assessment is required or not. The reason is, with global error assessment, the reported solution is from the subsidiary integration which is believed to be more accurate.

Velocity residuals. Unlike the two previous approaches, this technique works over the velocity space. So it is an indirect measure of particle tracing accuracy. The underlying idea is that an approximated solution of a problem can be always regarded as the exact solution of a different problem.

As mentioned before, the sampler embodied by the numerical solver provides not only particle positions at the sampling times but also approximates in the same way the first derivatives. So we define the velocity residual r at each sample point as the difference between this first derivative and the velocity obtained through the modelling interpolator at that position. Hence, the velocity residual is described as

$$r = \frac{d\,p}{d\,t} - \vec{V}(p)\,. \tag{5.8}$$

We regard this technique as a comparison between the interpolation carried out at the sampling process and the one at the modelling stage. It is particularly valuable if one can provide highly accurate interpolation at the modelling stage.

Data synchroniser. The aim of this filter is to ensure that data down the pipeline will be suitable for visualization. As mentioned before, there is no guarantee that reported data will be defined at exactly the same times as in the main trace.

The main trace $\mathcal{P}$ is described by $(x,\, y,\, z)$ positions in space at constant interval times, say, the set of times $\mathcal{T}$. But the last position may not be at a regular time. For example a crucial position that must be always recorded is the one where the particle leaves the domain. Hence the last position is always included. $\mathcal{P}$ is the benchmark to compare against.

Results from re-integration and velocity residuals are required to be filtered in order to include only those whose times are explicitly available in the main trace.

In the case of global error estimators, we do not attempt to synchronise with the sample points, but rather leave them at the positions at which they were calculated. Otherwise we would introduce errors.

So data is delivered as follows:

$$\mathcal{P}:\ x,\ y,\ z,\ (t)\,, \quad t \in\ \mathcal{T} \cup \{t_{n+1}\}\,, \qquad t_{n+1} - t_n \leq \alpha\ , \tag{5.9}$$

$$\mathcal{Q}:\ x,\ y,\ z,\ (t)\,, \quad t \in [\, t_{q_i} \ldots t_{q_k}\,] \ \subseteq\ \mathcal{T}, \qquad \Delta t = \pm\alpha\ , \tag{5.10}$$

$$\mathcal{E}:\ x,\ y,\ z,\ |\,\epsilon_x\,|,\ |\,\epsilon_y\,|,\ |\,\epsilon_z\,|,\ (t)\,, \quad t \in [\, t_{e_0} \ldots t_{e_k}\,] \ \subseteq\ \mathcal{T}, \tag{5.11}$$

$$\mathcal{R}:\Delta v_x,\ \Delta v_y,\ \Delta v_z,\ (t)\,, \quad t \in [\, t_{r_0} \ldots t_{r_k}\,] \ \subseteq\ \mathcal{T}, \qquad \Delta t = \alpha\ , \tag{5.12}$$

where $\mathcal{T} = [\, t_0 \ldots t_n\,]$, with $\Delta t = \alpha$ as the sampling frequency. $\mathcal{P}$, $\mathcal{Q}$, $\mathcal{E}$ and $\mathcal{R}$ are respectively the main particle trace, re-integration, global error estimators and velocity residual solutions.

5.2.5 Visualization

Having generated the accuracy data, the next step is to convert it into geometries which then are rendered. Let us substantiate the paradigms of visualization.

First, to convey accuracy we still use the same picture that shows the main trace. Both data and accuracy data will be superimposed. Second, a range of options are

offered to users, even allowing them to enhance the final image. Indeed visual perception varies amongst people. These options are:

– scaling primitives, particularly useful when differences of scale between accuracy data and main data are considerable;

– selection of accuracy data to display and which(s) visual paradigm(s) to be used;

– adjust visual attributes such as transparency;

– query data and then obtain textual feedback. It is helpful when the accuracy data is very small relative to the scale of the particle tracing.

To show the particle tracing, as $\mathcal{P}$ of (5.9), we use the common paradigm: a sphere is placed at each particle position and interconnected by straight-lines. If consecutive spheres imply constant variation in time, then one can infer the speed of the motion of the particle; the further apart the faster the particle moves.

In case of re-integration, as $\mathcal{Q}$ of (5.10), the aim is basically to show different positions of the particle, so distance and direction are the targets. In case of global error estimators, $\mathcal{E}$ of (5.11), the goal is to show the magnitude of the error along each $x-, y-, z-$ direction. Finally for velocity residuals $\mathcal{R}$ of (5.12), the target is difference in velocities. Notice that the data space here is different from the particle position of $\mathcal{P}$.

The first visualization metaphor is based on a *cuboid* aligned in the $x-, y-, z-$ directions, with edges proportional to the accuracy data in the related direction. So variations per axis are revealed. In a second one, a line or a *spike* is drawn straight from the reference point (000) of the cuboid to the opposite point (111). This is the classic example of showing two different positions. Both of these two techniques convey a discrete domain. In order to extend them into a continuous domain the two counterparts are: *tube* and *strip*. Figure 5.4 shows all the five visual paradigms mentioned. The radius of the tube is computed as indicated in Figure 5.5.

Finally, a note on the implementation. The visualization primitives are created as Open Inventor nodes that then are passed on for rendering to the IRIS Explorer envi-

ronment. So the render facilities of this environment can always be used, allowing for instance the change of the drawing style (solid to wireframe, *etc.*).

5.3 Tests

This section offers an insight into some tests we carried out. They are based on a realistic representation of the air flow between two parallel sheets of glass — one cold, one warm — a simulation of the gap in double glazing. The data set has been obtained from the solution of the Navier-Stokes equation using multi-grid code. It is defined over a Cartesian grid of dimension $[\,18 \times 10 \times 10\,]$.

For all the tests the velocity interpolation has been done using a trilinear interpolant. The Figures 5.6 to 5.10 present some results. The particles tend to travel near the boundaries showing a well defined trajectory for the circulation of the air. In the middle the motion is much slower.

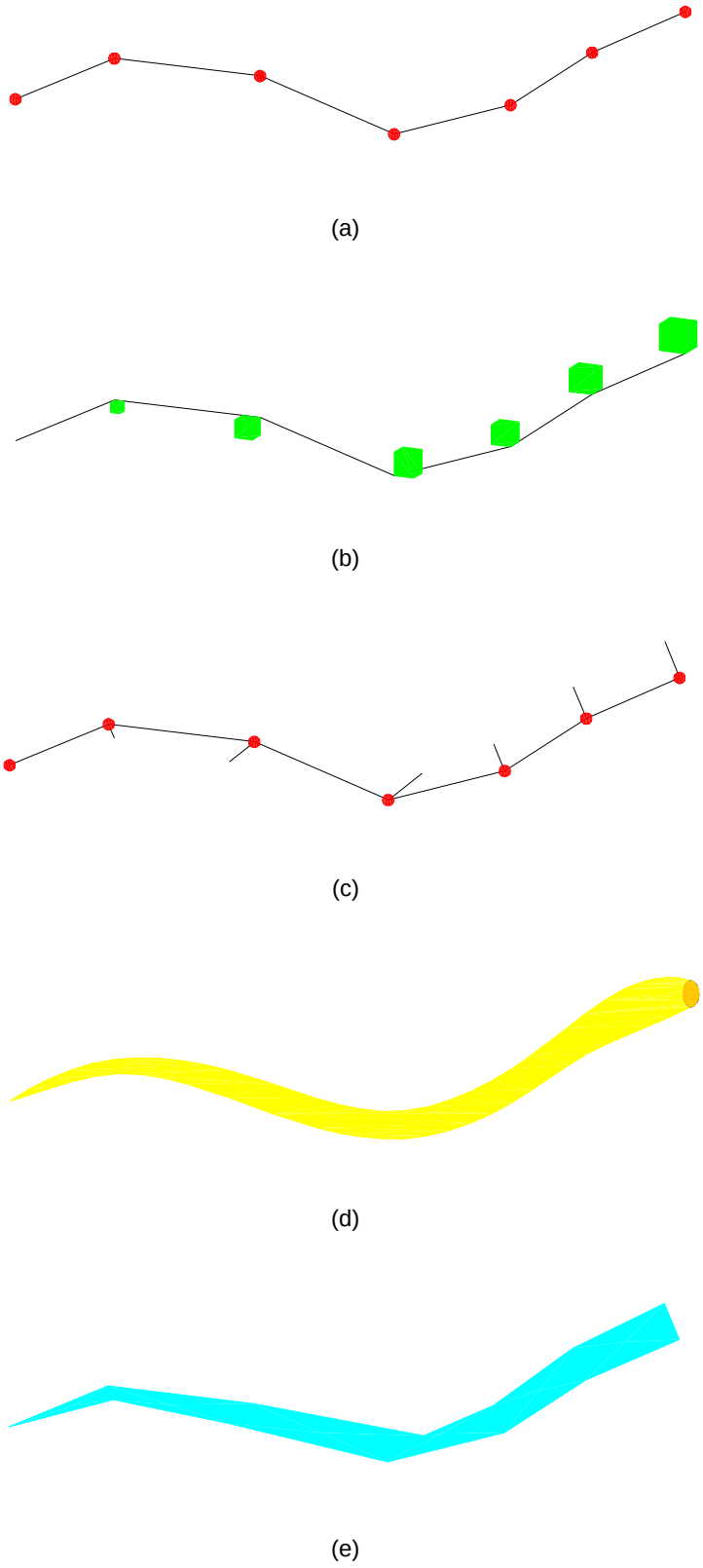


Figure 5.4: Visual paradigms. The main trace is depicted on (a). The accuracy data is convey such as (b) cuboides, (c) spikes, (d) tube or (e) strip.

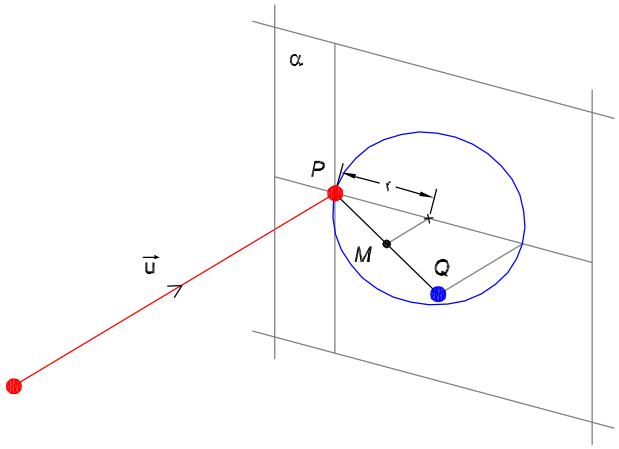
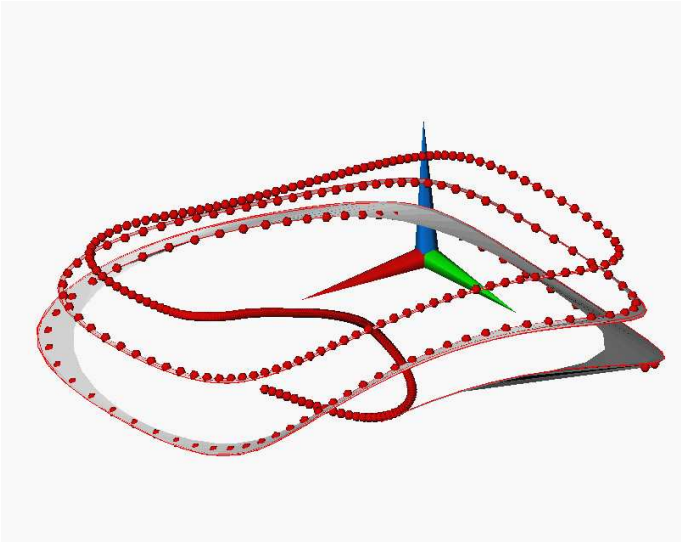
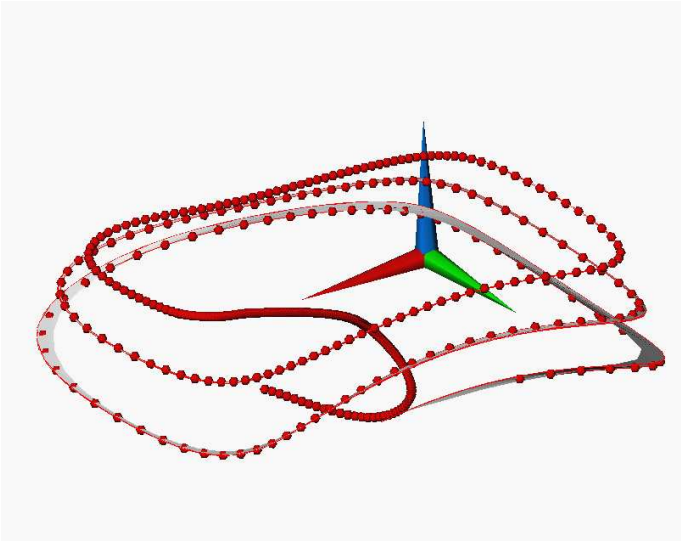


Figure 5.5: Definition of the tube. The cross-section of the tube at some point of the main trace is as follows: the direction of the trace $\vec{u}$ in that chunk defines a projection plane α. P is the particle position and Q is the ”assumed” accurate position. The mid-point M is then projected on the plane α and is going to be the centre of the cross-section. The radius is therefore the distance between P and that centre. Once that is defined, a number of points along the circle are computed (for example iteratively using polar coordinates). These points and similar ones in a previous cross-section allow to construct one chunk of the tube. As a final note one can consider defining the plane α on the basis of the previous and the next particle positions. Tests we made however suggest that it does not work smoothly when there are big changes in direction.



(a)



(b)

Figure 5.6: Sensitivity to re-integration, within a specific time interval. The solver uses RK 2(3) to compute the particle trace. The relative error tolerance on (a) is 0.008 whereas on (b) is 0.005. The geometry of the grey strip indicates differences in results between four re-integrations. In each of the examples above, differences in results are worth to note only from the first to the second integration (relative error tolerance reduced by 1/10).

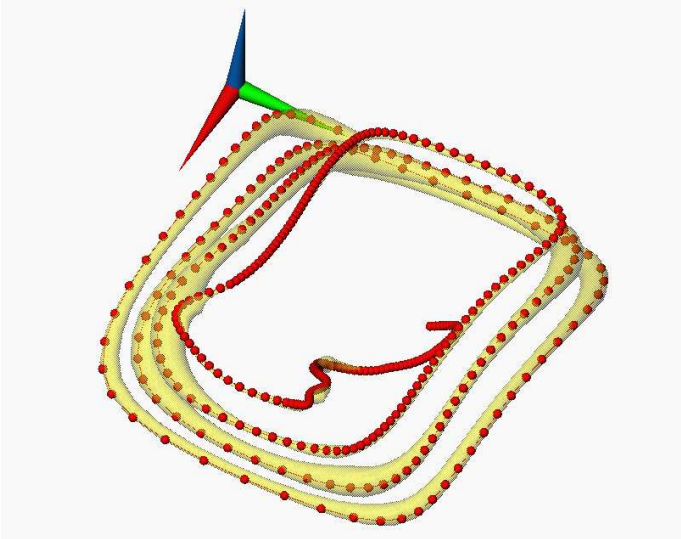


Figure 5.7: Trace of one particle (red sphere) within a specific time interval. The solver uses RK 4(5) and relative error tolerance as 0.008. The transparent yellow tube depicts a volume where the particle could lie, in the light of results of a second but likely to be a more accurate integration (relative error tolerance reduced to 0.0008).

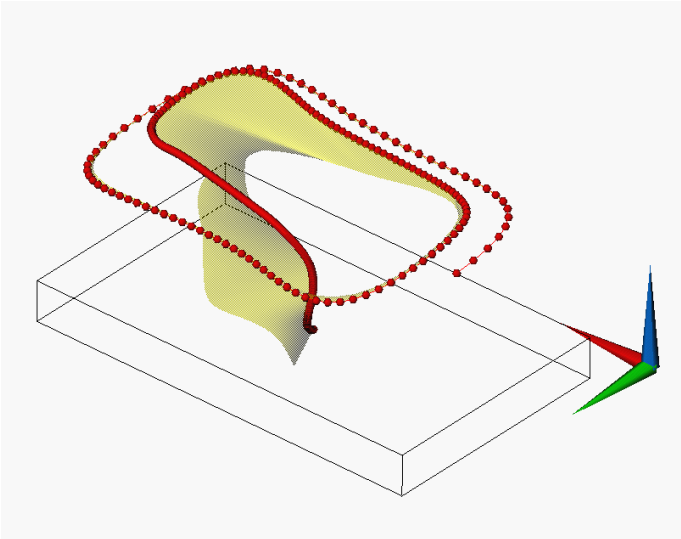
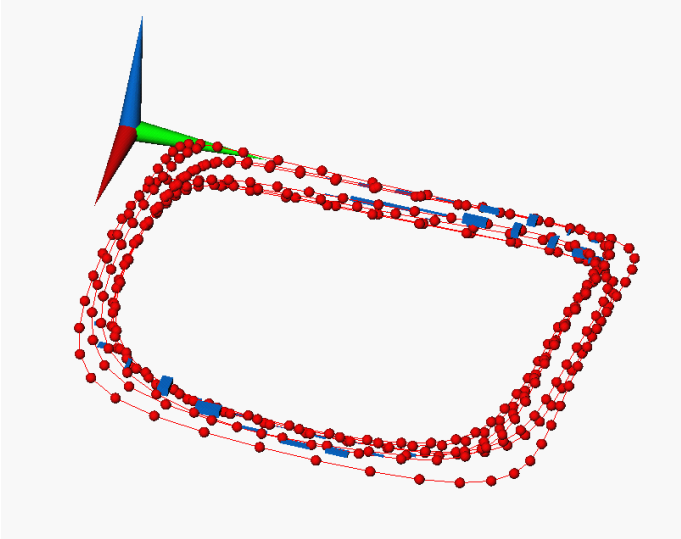
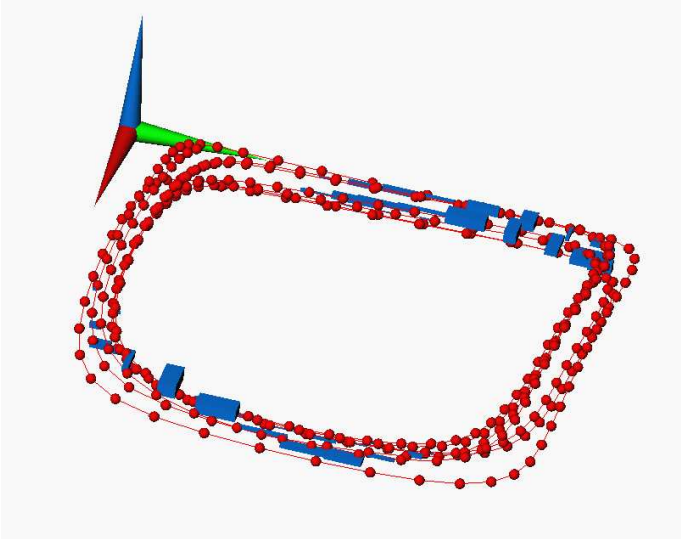


Figure 5.8: Trace of one particle (red sphere) within a given time interval. The solver uses RK 4(5) and relative error tolerance as 0.008. A second integration is performed backwards. The differences in both results are depicted through a transparent yellow strip.

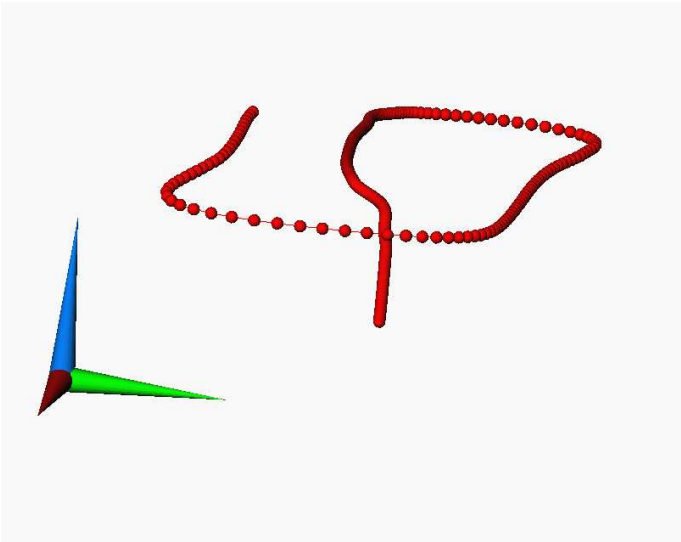


(a)

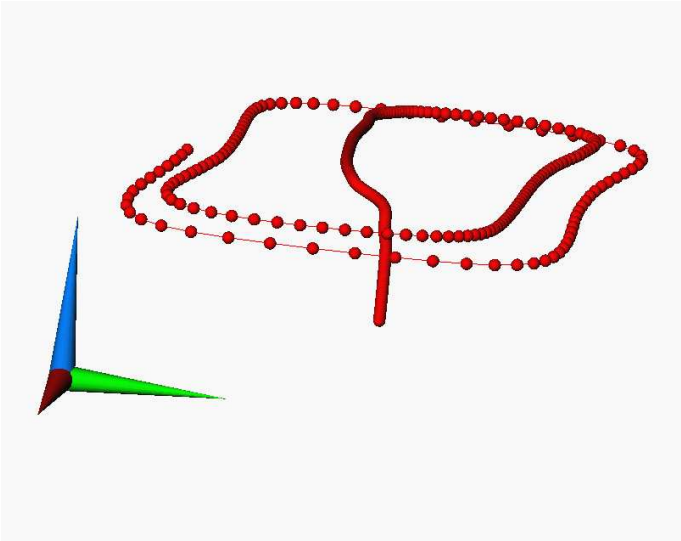


(b)

Figure 5.9: Trace of one particle (red sphere) within a given time interval. The solver uses RK 4(5) and relative error tolerance as 0.008. On (a) the velocity residuals are scaled by a factor of 30 whereas on (b) by a factor of 50. The visualization is misleading; the use of volume gives a false visual cue if these residuals are scaled and then compared against to each other. We also recall that the accuracy data does not indicate differences in particle positions but differences in velocities.



(a)



(b)

Figure 5.10: Global error assessment in one particle trace within a given time interval. The solver uses RK 4(5) and relative error tolerance as 0.008. On (a) the integrator stops at some point. It can not provide global error estimators from then onwards. Besides, when they are delivered, its magnitude matches the tolerance established, so in order to visualise them one has to scale the estimators. On (b) it is shown the trace when no global error assessment is asked.

5.4 Conclusions and Future Development

In this Chapter we have studied methods to show the accuracy of particle traces. In the light of the tests mentioned and others we carried out, we draw the following conclusions:

- By solving the same problem in a different way, one can obtain completely different results. For example, in the case of re-integration it is not even necessary to change the formulae; only a small change in the relative error tolerance is enough.
- The sensitivity to relative error tolerance with this solver is particularly noticeable when reduced by 1/10. There are no substantial gains in accuracy if re-integration is done with reductions by 1/100 or more.
- Global error estimators provided by this solver are difficult to obtain in practice. This may be due to the way the technique was implemented. Indeed, its computation is stopped if it is no longer credible for the relative tolerance established. This just shows how important it is for the user the understanding of a particular technique and the implementation issues involved.
- The velocity residuals technique allows comparison between interpolation from the sampler, and the interpolation from the data field. But in order to draw any relevant conclusions, we should have had a very accurate velocity interpolator in the field, which was not the case. Indeed, the error order of the interpolant of the sample points is even higher than the trilinear interpolant used.

As far as research directions are concerned we need to point out the error analysis at the modeling stage. One way of proceeding is to solve the same problem but with two different modelling interpolators. Then compare both results. We have indeed carried out some tests using the trilinear interpolant and the nearest neighbour technique (the value interpolated within a cell is of the nearest cell vertex). The differences quite expectedly were remarkable; the neighbour technique is simply poor. Instead tests should now concentrate on using an higher order interpolant to compare with the trilinear interpolant.

Nevertheless one should be aware of the major difficulty that derives from an higher-order interpolant: high computational cost as referred to in Section 5.1.3 on page 130.

Another aspect worth studying is how different modelling interpolators affect the RK integrator at any step. This will require the code of the numerical solver to be adjusted. So far we have seen the numerical solver (from NAG C library) as a ”black-box”.

Finally, tests involving user perception are required. This task validates (positively or negatively) the visual techniques. For example, as suggested in Figure 5.9, scaling the accuracy data and then representing it as cuboids gives a false visual clue. Also, querying data is a point to be focussed on. New techniques can be added if the user feedback demands to do so.

Chapter

SIX

Conclusions and Future Work

The only certainty is that nothing is certain.

In this final Chapter we aim to summarise the major aspects of the research undertaken, and to indicate some research directions. The order follows the outline of the thesis: first contouring, then isosurfacing and finally particle tracing.

If a major lesson has to be learnt from the work undertaken, it is that looking at the issue of accuracy in scientific visualization, not only leads to a better understanding of the techniques involved, and may unfold surprises to users; it also has the potential to point out further improvements.

6.1 Contouring

The work on contouring has shown us that techniques which are rather old are still worth studying. Indeed, being old does not necessarily means useless, or out of date; there is always space for innovation.

After a complete understanding of all the aspects involved, we started by looking at ways to obtain metrics for the error committed in classic methods. In these methods,

the contour was represented by straight lines instead of a hyperbolic curve, since at each rectangular cell the aim was to represent a bilinear function. That was the case in the basic method, using the asymptotic decider criterion to solve ambiguity inherent to the bilinear function, and the four triangles method.

During this process we realised that we could approximate each hyperbolic arc as two straight line segments. This conclusion is rather trivial; the four triangles technique already use it, although in that technique not all the end points of the line segments lie on the hyperbolic arc, nor it does always provide correct topology. In our method, the common point joining the two segments — the shoulder point — presents an interesting feature: it lies on the line joining the midpoint of the chord between the end points of the hyperbolic arc, and the saddle point of the bilinear function. We mathematically proved that was the case. As a result, we presented a new method that:

(a) provides correct topology in respect to the bilinear function, since we use the asymptotic decider criterion to solve eventual ambiguity, as the basic method does. Positional continuity between cells is therefore guaranteed;

(b) is more accurate in comparison to the other two methods. Comparing to the basic method, just the fact we use two line segments instead of one, supports the claim. In comparison to the four triangles, although both use the same number of line segments we note that all the end-points in our method lie on the hyperbolic arc and that does not happen in the four triangles method. Besides, the four triangles technique does not always provide correct topology so it is inaccurate per definition.

Concluding, the new method draws contour lines on quadrangular grids, with a bilinear function as reference for each cell. At no big expense of computational time, we improved substantially the accuracy of basic contouring methods.

Yet, no matter how far we go to improve the accuracy of solutions, there will be always uncertainty. We have established an exact error metric: the area whose boundary is defined by the contour solution and the representation of its approximation. Then, for each grid cell we mapped this value using a metaphor of ”dust cloud”: random points are plotted in the vicinity of the contour drawn, towards the exact contour. Another suggestion

made concerns the use of a solid but transparent background, in accordance to the metric value at each grid cell.

6.2 Isosurfacing

Isosurfacing is the natural extension of contouring. We strictly focused our attention to Marching-Cubes, which is by far the most well known method. This feature was the natural reason for us to choose it as our reference. As far as the method goes, there are two important research landmarks apart from its presentation in 1987, by Lorensen and Cline:[67] the use of the asymptotic decider to solve ambiguities in the faces of the cube, by Nielson and Hamann in 1991;[78] and the extension of this criterion into the interior of the cube, in 1994, by Natarajan.[76]

As we have done in contouring we initially paid particular attention to the study of the technique. The effort was rewarded. Indeed, the story was not completely understood as far as MC configurations were concerned. First, we reported a new subcase. Second, we proved mathematically that some of subcases already presented were impossible to happen. It is noteworthy that a subsequent review of the method was discussing subcases that we now rule out.[111]

Another conclusion we drew was that although the problem of tunnels in the interior of the cube was correctly discussed by Natarajan, assuming trilinearity within the cube, not enough attention was devoted to the way triangulation was carried out. The problem is that a surface has to be continuous and not self intersecting, and therefore no triangles are allowed to lie in the faces of the cube. As an additional remark, we were not able to produce tunnels in one MC configuration (13), as suggested by Natarajan.

Considering the work on contouring and the understanding of the MC method, we developed a more accurate Marching-Cubes method. The aim was to represent as accurately as possible the trilinear interpolant within each cell. Similarly to the normal MC, it was guaranteed positional continuity of the isosurface across adjacent cells, and therefore correct topology in the faces of each cell.

This accurate Marching-Cubes relies mostly on supplementary points on the iso-

surface, either lying in the faces, or in the interior of the cell. In respect to the faces of the cube, we used the findings on contouring: the supplementary point we called a face shoulder point resembles the shoulder point in contouring; and the asymptotic decider is used to solve eventual ambiguities in the faces. For the interior of the cube, our strategy was based on the observation that a cubic cell can be pictured as a range of many two dimensional slices. In other words, the isosurface can be seen as a pile of consecutive parallel contours. These supplementary points in the interior are inflection points — points such that two first derivatives of the trilinear interpolant vanish — or else bi-shoulder points — points that are simultaneously shoulder points on the contour of the trilinear interpolant on two orthogonal slices through the cube. An inflection point reflects the topology in the interior of the cell while a bi-shoulder point is more linked to accuracy issues, without affecting the way topology is defined.

A crucial task in isosurfacing is the accomplishment of triangulation. In comparison to MC, our method takes a different view. Rather than following configurations as the MC does, we are driven by topological polygons, no matter the MC configuration from which they arise. The methodology is as follows: we first establish the topological polygons, using the asymptotic decider to solve ambiguities, and including face shoulder points. Then we consider that the triangulation of polygons falls into three situations: polygons lying on loop-back faces, polygons not lying on loop-back faces, and situations of tunnels in the interior of the cell. In the first case, the triangulation uses inflection points. In the second case, one bi-shoulder point (there are no inflection points defined) and in the latter one, again inflection points are used. This last case presents interesting features. First a tunnel is formed upon two topological polygons not lying on loop-back faces. Second, there are six inflection points, by definition the maximum number. In that respect, we noticed that if there are six inflection points then all are located in the corners of a cuboid. This result is the foundation for the triangulation of tunnels. A polyline is drawn linking all the inflection points along the edges of the cuboid. Consequently each polygon is triangulated independently, being the polyline mentioned as the ”border” between the two polygons.

In conclusion, the use of supplementary points helped:

– to define correct topologies;

– to deliver more accurate solutions in comparison to MC;

– to make triangulation more systematic.

We are also aware that an alternative way to represent the trilinear interpolant to greater accuracy is to pursue a crude but simple approach — namely, to subdivide each cube into many smaller cubes, and applying the MC algorithm within each minor cube. However the aim of our approach has been to use a more ”intelligent” approach where we try to represent the fundamental topology of the trilinear interpolant within the original cube, with a minimum of additional points. This has the significant advantage of course in practice of not generating an inordinate number of triangles.

Now let us turn to ideas for further developments. One aspect that deserves more study involves the bi-shoulder points. First their computation is not easy to perform since it requires numerical approximation. Besides, one can obtain more than one bi-shoulder point within a topological polygon (that do not lie on loop-back faces). This may raise some question marks since we use just one in the triangulation. It seems that triangulation of polygons not lying on loop-back faces may merit further improvements.

We should also quantify the additional computational cost involved with this new method. At the moment we believe it is affordable; the number of triangles produced is automatically compared to the traditional MC. But the increasing number of triangles may be of major concern. That is the price to pay for accurate solutions. However, we suggest that if the number is not affordable then the question of decimation of triangles should be considered. Actually, we can use the information generated at the time of the definition of topological polygons to speed up the decimation. For instance, in case of triangulation of topological polygons that do not lie on loop-back faces, two adjacent triangles that have a face shoulder point as common vertex can be merged if the angle due to that shoulder point implies the triangles are nearly flat.

Also, we should consider to use techniques that avoid the processing of cells that do not contribute to the isosurface.

Similarly to contouring, we also presented some ideas to show accuracy but no

implementation was carried out. Namely, to use an error metric as the one in contouring, and then to map it by means of texture in the surface drawn. Another idea seems to be more interesting, since it suggests that we can go further as far as accurate solutions are concerned: for each triangle drawn we compute another point on the surface, close to the triangle. Notice that the vertices of the triangle are points on the surface. Then accuracy can be depicted by drawing lines from that extra point to the vertices of the triangle. If one is not happy with the accuracy of our method then they can replace the triangle by three new triangles, now involving the three old vertices plus the extra point. And the process can even be recursive.

One aspect we will study in the very near future is the representation of interval volumes. That is, rather than showing an isosurface $F = \Theta$, the objective is to show an interval volume such as $\alpha \leq F \leq \beta$. We believe that something interesting can be achieved in this topic, based on the accurate Marching-Cubes.

6.3 Particle Tracing

In respect to particle tracing, we have presented ways to show the accuracy of the traces computed. Unlike the two previous techniques no effort was made to improve the technique. The emphasis was just on giving users an indication of the error committed, which anyway has required a deep understanding of the technique. The particle tracing method used as reference was based on the one found in the NAGAdvectSimple module from IRIS Explorer. We have used a very reliable Runge-Kutta integrator from the NAG library, with adaptive time-stepping, and a trilinear interpolator to obtain values from the velocity data set.

In a first stage we presented three strategies to measure the accuracy involved in the integration step: re-integration, global error estimators, and velocity residuals. All the ideas were implemented following an architecture we established, denoting the dataflow paradigm. The fate of these strategies was completely different. Velocity residuals is an indirect measure of accuracy. Besides, in order to be properly used, we should consider interpolators of higher accuracy order. Global error estimators was the technique most

theoretically acceptable but the estimators were difficult to obtain in practice with this solver. In the end, re-integration was the most successful technique. Indeed it has shown how different the trace of particles for the same problem can be. This is something users should be aware of. From the variants we set up, special reference goes to the one where the relative error tolerance was reduced, by a factor of ten.

Another aspect worth to mention is related to the visual paradigms used. First of all both the trace of particle and its accuracy were conveyed in the same picture. Among the techniques shown, the more successful were the tube and the strip. In both cases the underlying goal was to depict all the possible solutions for a particle trace. Anyway, further research should involve more user perception analysis.

Proceeding with the issue of research directions, we emphasise the study of the error committed at the modelling stage. This involves the analysis of different interpolators, and how they affect the integration. This poses a problem in the use of the numerical solver, which must not be seen as a black-box as we so far have considered.

BIBLIOGRAPHY

[1] Chandrajit Bajaj, Valerio Pascucci, and Daniel Schikore. Fast isocontouring for improved interactivity. In *Proceedings of 1996 Symposium on Volume Visualization*, pages 39–46, October 1996.

[2] Bahari Belaton. *A Model Based Approach to Scientific Visualization*. PhD thesis, School of Computer Studies, University of Leeds, United Kingdom, September 1995.

[3] R Daniel Bergeron and Arie Kaufman, editors. *IEEE Visualization 94 Conference Proceedings*, Washington, D.C., USA, October 1994. IEEE Computer Society, IEEE Computer Society Press.

[4] J Bertin. *Seminology of Graphics*. University of Wisconsin Press, Madison, 1983.

[5] James Blinn. Light reflection functions for simulation of clouds and dusty surfaces. *Computer Graphics*, 16(3):21–29, 1982. ACM SIGRAPPH Conference Proceedings.

[6] R Brankin, I Gladwell, and L Shampine. Rksuite: A suite of explicit runge-kutta codes for the initial value problem for odes softreport 91-s1. Technical report, Department of Mathematics, Southern Methodist University, Dallas, TX 75275, USA, 1991.

[7] Manfred Brill, Hans Hagen, Hans-Christian Rodrian, Wladimir Djatschin, and Stanislav Klimenko. Streamball techniques for flow visualization. In Bergeron and Kaufman,[3] pages 225–231.

[8] Ken Brodlie, editor. *Mathematical Methods in Computer Graphics and Design*. Academic Press, London, 1980.

[9] Ken Brodlie. A classification scheme for scientific visualization. In Earnshaw and Watson,[31] chapter 8, pages 125–140.

[10] Ken Brodlie. A typology for scientific visualization. In Hearnshaw and Unwin,[45] pages 34–41.

[11] Ken Brodlie, Lesley Carpenter, Rae Earnshaw, Jullian Gallop, Roger Hubbold, Anne Munford, Chris Osland, and Peter Quarendon, editors. *Scientific Visualization — Techniques and Applications*. Springer-Verlag, Berlin, 1992.

[12] Steve Bryson and Creon Levit. The virtual windtunnel: an environment for the exploration of three-dimensional unsteady fluid flows. *IEEE Computer Graphics & Applications*, 12(4):25–34, July 1992.

[13] Pieter Bunning. Sources of error in the graphical analysis. *Journal of Scientific Computing*, 3(2):149–164, 1988.

[14] Brian Cabral and Leith Leedom. Imaging vector fields using linear integral convolution. *IEEE Computer Graphics & Applications*, pages 263–270, 1993. ACM SIGGRAPH Conference Proceedings.

[15] S Chan and E Purisima. A new tetrahedral tesselation scheme for isosurface generation. *Computer & Graphics*, 22(1):83–90, 1998.

[16] Lih-Shyang Chen, Gabor Herman, R Anthony Reynolds, and Jayaram Udupa. Surface shading in the cuberrile environment. *IEEE Computer Graphics & Applications*, 5(12):33–43, December 1985.

[17] Paolo Cignoni, Paola Marino, Claudio Montani, Enrico Puppo, and Roberto Scopigno. Speeding up isosurface extraction using interval trees. *IEEE Transactions on Visualization and Computer Graphics*, 3(2):158–170, April-June 1997.

[18] William Cleveland. *Visualizing Data*. Hobart Press, Summit, 1993.

[19] H Cline, W Lorensen, S Ludke, C Crawford, and B Teeter. Two algorithms for the three-dimensional construction of tomograms. *Medical Physics*, 15(3):320–327, June 1988.

[20] Brian Collins. Data visualization — has it all been seen before? In Earnshaw and Watson,[31] chapter 1, pages 3–28.

[21] Roger Crawfis and Nelson Max. Texture splats for 3d scalar and vector field visualization. In Nielson and Bergeron,[77] pages 261–266.

[22] Patricia Crossno and Edward Angel. Isosurface extraction using particle systems. In Yagel and Hagen,[128] pages 495–498.

[23] Dave Darmofal and Robert Haimes. An analysis of 3-d particle path integration algorithms. *Journal of Computational Physics*, 123:182–195, 1995.

[24] Willem de Leeuw and Jarke van Wijk. A probe for local flow field visualization. In Nielson and Bergeron,[77] pages 39–45.

[25] Willem de Leeuw and Jarke van Wijk. Enhanced spot noise for vector field visualization. In Nielson and Silver,[80] pages 233–239.

[26] Robert Debrin, Loren Carpenter, and Pat Hanraham. Volume rendering. *Computer Graphics*, 22(4):65–74, 1988.

[27] Thierry Delmarcelle and Lambertus Hesselink. Visualization of second order tensor fields and matrix data. In Kaufman and Nielson,[56] pages 316–323.

[28] Thierry Delmarcelle and Lambertus Hesselink. Visualizing second-order tensor fields with hyperstreamlines. *IEEE Computer Graphics & Applications*, 13(4):25–33, 1993.

[29] Thierry Delmarcelle and Lambertus Hesselink. A unified framework for flow visualization. In Lawrence Rosemblum, Rae Earnshaw, José Encarnaç ao, Hans Hagen, Arie Kaufman, S Klimenko, Gregory Nielson, Frits Post, and Daniel Thalmann, editors, *Scientific Visualization — Advances and Challenges*, chapter 5, pages 129–170. Academic Press, 1995.

[30] Martin Dürst. Letters: Additional reference to ”marching cubes”. *Computer Graphics*, 22(2):72–73, April 1988.

[31] Rae Earnshaw and David Watson, editors. *Animation and Scientific Visualization — Tools and Scientific Applications*. Academic Press, London, 1993.

[32] A Ekoule, F Peyrin, and C Odet. A triangulation algorithm from arbitrary shaped multiple planar contours. *ACM Transactions on Graphics*, 10(2):182–199, April 1991.

[33] Gregory Fletcher. *Advances in Volume Visualization*. PhD thesis, School of Computer Studies, University of Leeds, United Kingdom, 1994.

[34] James Foley, Andries van Dam, Steven Feiner, and John Hughes. *Computer Graphics — Principles and Practice*. The Systems Programming Series. Addision-Wesley, 2nd edition, 1990.

[35] L Forssell and S Cohen. Using line integral convolution for flow visualization: Curvilinear grids, variable-speed and unsteady flows. *IEEE Transactions on Visualization and Computer Graphics*, 1(2):133–141, 1995.

[36] Lisa Forssell. Visualizing flow over curvilinear grid surfaces using line integral convolution. In Bergeron and Kaufman,[3] pages 240–247.

[37] Henry Fuchs, Zvi Kedem, and Samuel Uselton. Optimal surface reconstruction from planar contours. *Communications of the ACM*, 20(10):693–702, October 1977. ACM SIGGRAPH Conference Proceedings.

[38] J Gibson. *The Perception of the Visual World*. Houghton Mifflin Co., Boston, USA, 1950.

[39] Fernando Gil and Helder Macedo. *Viagens do Olhar — Retrospeccao, Visao e Profecia no Renascimento Português*. Campo das Letras, Porto, 1998.

[40] M Goodchild, B Buttenfield, and J Wood. Introduction to visualizating data validity. In Hearnshaw and Unwin,[45] pages 141–149.

[41] D Gordon and R Reynolds. Image space shading of 3-dimensional objects. *Computer Graphics & Image Processing*, 29(3):361–376, March 1985.

[42] A Guéziec and R Hummel. Exploiting triangulated surface extraction using tetrahedral decomposition. *IEEE Transactions on Visualization and Computer Graphics*, 1:328–342, 1995.

[43] Robert Haber and David McNabb. Visualization idioms: a conceptual model for scientific visualization systems. In B Shriver, G Nielson, and L Rosenblum, editors, *Visualization in Scientific Computing*, pages 74–93. IEEE Computer Society Press, Los Alamitos, 1990.

[44] Mark Hall and Joe Warren. Adaptive polygonization of implicitly defined surfaces. *IEEE Computer Graphics & Applications*, 10:33–42, November 1990.

[45] H Hearnshaw and D Unwin, editors. *Visualization in Geographical Information Systems*. John Wiley & Sons, 1994.

[46] James Helman and Lambertus Hesselink. Visualizing vector field topology in fluid flows. *IEEE Computer Graphics & Applications*, 11(3):36–46, 1991.

[47] G Herman and H Liu. Three-dimensional display of human organs from computed tomograms. *Computer Graphics and Image Processing*, 9:1–21, 1979.

[48] F Robert Hopgood, David Duce, Julian Gallop, and Dale Sutcliffe. *Introduction to the Graphical Kernel System (GKS)*. Number 19 in A.P.I.C. Studies in Data Processing. Academic Press, London, 1983.

[49] J Hultquist. Constructing stream surfaces in steady 3d vector fields. In Kaufman and Nielson,[56] pages 171–178.

[50] Vitoria Interrante and Chester Grosch. Strategies for effectively visualizing 3d flow in volume lic. In Yagel and Hagen,[128] pages 421–424.

[51] T Itoh and K Koyamada. Automatic isosurface propagation using an extrema graph and sorted boundary cell lists. *IEEE Transactions on Visualization and Computer Graphics*, 1(4):319–327, December 1995.

[52] Bruno Jobard and Wilfrid Lefer. The motion map: Efficient computation of steady flow animations. In Yagel and Hagen,[128] pages 323–328.

[53] Mark Jones. The production of volume data from triangular meshes using voxelisation. *Computer Graphics forum*, 15(5):311–318, 1996.

[54] Mark Jones and Min Chen. A new approach to the construction of surfaces from contour data. *Computer Graphics forum*, 13(3):75–84, September 1994. Eurographics Conference Proceedings.

[55] Mark Jones and Min Chen. Fast cutting operations on three dimensional volume datasets. In *Visualization in Scientific Computing*, pages 1–8. Springer-Verlag, January 1995.

[56] Arie Kaufman and Gregory Nielson, editors. *IEEE Visualization 92 Conference Proceedings*, Boston, Massachusetts, USA, October 1992. IEEE Computer Society, IEEE Computer Society Press.

[57] David Kenwright and David Lane. Optimization of time-dependent particle tracing using tetrahedral decomposition. In Nielson and Silver,[80] pages 321–328.

[58] David Kenwright and Gordon Mallinson. A 3-d streamline tracking algorithm using dual stream functions. In Kaufman and Nielson,[56] pages 62–68.

[59] E Keppel. Approximating complex surfaces by triangulation of contour lines. *IBM Journal Research Development*, (3):2–11, 1975.

[60] Ming-Hoe Kiu and David Banks. Multi-frequency noise for lic. In Yagel and Nielson,[129] pages 121–126.

[61] David Knight and Gordon Mallinson. Visualizing unstructured flow data uusing dual stream functions. *IEEE Transactions on Visualization and Computer Graphics*, 2(4):355–363, 1996.

[62] Mark Levoy. Volume rendering — display of surfaces from volume data. *IEEE Computer Graphics & Applications*, 8(3):29–37, May 1988.

[63] Yarden Livnat, Han-Wei Shen, and Christopher Johnson. A near optimal isosurface extraction algorithm using the span space. *IEEE Transactions on Visualization and Computer Graphics*, 2(1):73–84, March 1996.

[64] Suresh Lodha, Alex Pang, Robert Sheehan, and Craig Wittenbrink. Uflow: Visualizing uncertainty in fluid flow. In Yagel and Nielson,[129] pages 249–254.

[65] Adriano Lopes and Ken Brodlie. Accuracy in 3d particle tracing. In Hans-Christian Hege and Konrad Polthier, editors, *Mathematical Visualization*, pages 329–341. Springer-Verlag, Heidelberg, 1998.

[66] Adriano Lopes and Ken Brodlie. Accuracy in contour drawing. In Eurographics UK, editor, *Eurographics UK 98 Conference Proceedings*, pages 301–311, Leeds, March 1998. University of Leeds.

[67] William Lorensen and Harvey Cline. Marching cubes: A high resolution 3d surface construction algorithm. *Computer Graphics*, 21(4):163–169, July 1987. ACM SIGGRAPH Conference Proceedings.

[68] Kwan-Liu Ma and Philip Smith. Virtual smoke: An interactive 3d flow visualization technique. In Kaufman and Nielson,[56] pages 46–53.

[69] E Marey. *La Méthod Graphique*. Paris, 1885.

[70] Nelson Max, Roger Crawfis, and Dean Williams. Visualizing wind velocities by advecting cloud textures. In Kaufman and Nielson,[56] pages 179–183.

[71] Bruce McCormick, Thomas DeFanti, and Maxine Brown. Visualization in scientific computing. *Computer Graphics*, 21(6):1–14, 1987.

[72] David Meyers, Shelley Skinner, and Kenneth Sloan. Surfaces from contours. *ACM Transactions on Graphics*, 11(3):228–258, July 1992.

[73] C Montani, R Scateni, and R Scopigno. Discretized marching cubes. In Bergeron and Kaufman,[3] pages 281–287.

[74] M Mortenson. *Geometric Modeling*. John Wiley & Sons, Inc., 1985.

[75] Vitor Murtinho. Perspectivas: O espelho maior ou o espaço do espanto. Master's thesis, Departamento de Arquitectura, F.C.T. Universidade de Coimbra, 1993.

[76] B Natarajan. On generating topologically consistent isosurfaces from uniform samples. *The Visual Computer*, 11:52–62, 1994.

[77] Gregory Nielson and R Daniel Bergeron, editors. *IEEE Visualization 93 Conference Proceedings*, San Jose, California, USA, October 1993. IEEE Computer Society, IEEE Computer Society Press.

[78] Gregory Nielson and Bernd Hamann. The asymptotic decider: Resolving the ambiguity in marching cubes. In Nielson and Rosemblum,[79] pages 83–90.

[79] Gregory Nielson and Larry Rosemblum, editors. *IEEE Visualization 91 Conference Proceedings*. IEEE Computer Society, IEEE Computer Society Press, 1991.

[80] Gregory Nielson and Deborah Silver, editors. *IEEE Visualization 95 Conference Proceedings*, Atlanta, Georgia, USA, October 1995. IEEE Computer Society, IEEE Computer Society Press.

[81] Paul Ning and Jules Bloomenthal. An evaluation of implicit surface tilers. *IEEE Computer Graphics & Applications*, 13(6):33–41, November 1993.

[82] J-M Oliva, M Perrin, and S Coquillart. 3d reconstruction of complex polyhedral shapes from contours using a simplified voronoi diagram. *Computer Graphics forum*, pages 397–408, August 1996. Eurographics Conference Proceedings.

[83] H Panofsky. Objective wheather-map analysis. *Journal of Meteorology*, 6(6):386–392, December 1949.

[84] Steven Parker, Peter Shirley, Yarden Livnat, Charles Hansen, and Peter-Pike Sloan. Interactive ray tracing for isosurface rendering. In David Ebert, Hans Hagen, and Holly Rushmeier, editors, *IEEE Visualization 98 Conference Proceedings*, pages 233–238, Research Triangle Park, North Carolina, USA, October 1998. IEEE Computer Society, ACM Press.

[85] Bradley Payne and Arthur Toga. Surface mapping brain function on 3d models. *IEEE Computer Graphics & Applications*, 10:33–41, September 1990.

[86] Frits Post and Theo van Walson. Fluid flow visualization. In Hans Hagen, Heinrich Müller, and Gregory Nielson, editors, *Focus on Scientific Visualization*, pages 1–40. Springer-Verlag, Berlin, 1993.

[87] William Reeves. Particle systems — a technique for modelling a class of fuzzy objects. *ACM Transactions on Graphics*, 2(2):91–108, 1983.

[88] William Reeves and Riki Blau. Approximate and probabilistic algorithms for shading and rendering structured particle systems. *Computer Graphics*, 19(3):313–322, 1985. ACM SIGGRAPH Conference Proceedings.

[89] Jonathan Roberts. *Aspects of Abstraction in Scientific Visualization*. PhD thesis, The University of Kent at Canterbury, United Kingdom, 1995.

[90] Arthur Robinson. The genealogy of the isopleth. *The Cartographic Journal*, 8(1):49–53, June 1971.

[91] Paolo Sabella. A rendering algorithm for visualizing 3-d scalar fields. *Computer Graphics*, 22(4):51–58, 1988.

[92] Malcolm Sabin. Contouring — the state of the art. In Rae Earnshaw, editor, *Fundamental Algorithms for Computer Graphics*, volume 17 of *NATO ASI Series F: Computer and Systems Sciences*, chapter 5, pages 411–482. Springer-Verlag, Berlin, 1985.

[93] W Schroeder, C Volpe, and W Lorensen. The stream polygon: A technique for 3d vector field visualization. In Nielson and Rosemblum,[79] pages 126–132.

[94] Will Schroeder, Ken Martin, and Bill Lorensen. *The Visualization Toolkit — An Object-Oriented Approach to 3D Graphics*. Prentice Hall PTR, 1996.

[95] L Shampine and H Watts. Global error estimation for ordinary differential equations. *ACM Transactions on Mathematical Software*, 2(2):172–186, June 1976.

[96] Lawrence Shampine and Ian Gladwell. The next generation of runge-kutta codes. In J Cash and I Gladwell, editors, *Computational Ordinary Differential Equations*, pages 145–164. Oxford University Press, 1992.

[97] Han-Wei Shen, Charles Hansen, Yarden Livnat, and Christopher Johnson. Isosurfacing in span space with utmost efficiency (issue). In Yagel and Nielson,[129] pages 287–294.

[98] Han-Wei Shen and David Kao. Uflic: A line integral convolution algorithm for visualizing unsteady flows. In Yagel and Hagen,[128] pages 317–322.

[99] D Stalling and H Hege. Fast and resolution independent line integral convolution. *Computer Graphics*, pages 249–256, 1995. ACM SIGGRAPH Conference Proceedings.

[100] J Stolk and J van Wijk. Surface-particles for 3d flow visualization. In Andrea Hin and Frits Post, editors, *Advances in Scientific Visualization*. Springer-Verlag, 1992.

[101] Walter Strauss, editor. *Woodcuts and wood blocks of Albrecht Dürer*. Abaris Books, New York, 1980.

[102] Dale Sutcliffe. Contouring over rectangular grids and skewed rectangular grids — an introduction. In Brodlie,[8] chapter 2, pages 39–62.

[103] Edward Tufte. *The Visual Display of Quantitative Information*. Graphic Press, Cheshire, Connecticut, 1983.

[104] Edward Tufte. *Envisioning Information*. Graphic Press, Cheshire, Connecticut, 1990.

[105] Edward Tufte. *Visual Explanations — Images and Quantities, Evidence and Narrative*. Graphic Press, Cheshire, Connecticut, 1997.

[106] John Tukey. *Exploratory Data Analysis*. Addison-Wesley, 1977.

[107] David Unwin. Geographical information systems and the problem of ”error and uncertainty”. *Progress in Human Geography*, 19(4):549–558, 1995.

[108] C Upson, T Faulhaber, D Kamins, D Schlegel, D Laidlaw, J Vroom, R Gurwitz, and A van Dam. The application visualization system: a computational environment for scientific visualization. *IEEE Computer Graphics & Applications*, 9(4):30–42, 1989.

[109] Craig Upson and Michael Keeler. V-buffer: Visible volume rendering. *Computer Graphics*, 22(4):59–64, August 1988.

[110] Frans van Der Wel, Rob Hootsmans, and Ferjan Ormeling. Visualization of data quality. In A MacEachren and D Taylor, editors, *Modern Cartography*, volume 2, chapter 16, pages 313–331. Elsevier Science Ltd, 1994.

[111] Allen van Gelder and Jane Wilhelms. Topological considerations in isosurface generation. *ACM Transactions on Graphics*, 13(4):337–375, October 1994.

[112] Jarke van Wijk. Spot noise: Texture synthesis for data visualisation. *Computer Graphics*, 25(4):249–256, 1991.

[113] Jarke van Wijk. Rendering surface-particles. In Kaufman and Nielson,[56] pages 54–61.

[114] Jarke van Wijk. Flow visualization with surface particles. *IEEE Computer Graphics & Applications*, 13(4):18–24, July 1993.

[115] Jarke van Wijk. Implicit stream surfaces. In Nielson and Bergeron,[77] pages 245–252.

[116] Jeremy Walton. Visualization benchmarking: A practical application of 3d publishing. In Eurographics UK, editor, *Eurographics UK 96 Conference Proceedings*, pages 339–351, London, March 1996. Imperial College of London.

[117] David Watson. *Contouring — A Guide to the Analysis and Display of Spatial Data*, volume 10 of *Computer Methods in the Geosciences*. Pergamon Press, Oxford, 1992.

[118] Rainer Weggenkittl and Edward Gröler. Fast oriented line integral convolution for vector field visualization via the internet. In Yagel and Hagen,[128] pages 309–315.

[119] Lee Westover. Footprint evaluation for volume rendering. *Computer Graphics*, 24(4):367–376, August 1990. ACM SIGGRAPH Conference Proceedings.

[120] Jane Wilhelms and Allen van Gelder. Octrees for faster isosurface generation. *ACM Transactions on Graphics*, 11:221–22, 1992.

[121] Craig Wittenbrink, Elijah Saxon, Jeff Furman, Alex Pang, and Suresh Lodha. Glyphs for visualizing uncertainty in environmental vector fields. In *SPIE & IS&T Conference Proceedings on Electronic Imaging: Visual Data Exploration and Analysis*, pages 87–100, February 1995.

[122] Pak Wong and R Daniel Bergeron. 30 years of multidimensional multivariate visualization. In Gregory Nielson, editor, *Scientific Visualization — Overviews, Methodologies, and Techniques*, chapter 1, pages 3–33. IEEE Computer Society Press, Los Alamitos, 1997.

[123] Jason Wood. *Collaborative Visualization*. PhD thesis, School of Computer Studies, University of Leeds, United Kingdom, 1998.

[124] Jason Wood, Ken Brodlie, and Helen Wright. Visualization over the world wide web and its application to environmental data. In Yagel and Nielson,[129] pages 81–86.

[125] Jason Wood, Helen Wright, and Ken Brodlie. Collaborative visualization. In Yagel and Hagen,[128] pages 253–259.

[126] Geoff Wyvill, Craig McPheeters, and Brian Wyvill. Data structure for *soft* objects. *The Visual Computer*, 2:227–234, 1986.

[127] Larry Yaeger, Craig Upson, and Robert Myers. Combining physical and visual simulation — creation of the planet jupiter for the film ”2010”. *Computer Graphics*, 20(4):85–93, August 1986. ACM SIGGRAPH Conference Proceedings.

[128] Roni Yagel and Hans Hagen, editors. *IEEE Visualization 97 Conference Proceedings*, Phoenix, Arizona, USA, October 1997. IEEE Computer Society, ACM Press.

[129] Roni Yagel and Gregory Nielson, editors. *IEEE Visualization 96 Conference Proceedings*, San Francisco, California, USA, October/November 1996. IEEE Computer Society, ACM Press.

[130] Yong Zhou, Weihai Chen, and Zesheng Tang. An elaborate ambiguity detection method for constructing isosurfaces within tetrahedral meshes. *Computer & Graphics*, 19(3):353–364, 1995.