THE USE OF INTERVAL-RELATED EXPERT KNOWLEDGE IN PROCESSING 2-D AND 3-D DATA, WITH AN EMPHASIS ON APPLICATIONS TO GEOSCIENCES AND BIOSCIENCES

ROBERTO ARAIZA

Department of Computer Science

APPROVED:

Vladik Kreinovich, Ph.D., Chair

Luc Longpré, Ph.D.

Martine Ceberio, Ph.D.

Pavel Šolín, Ph.D.

a mi madre

(otra vez)

THE USE OF INTERVAL-RELATED EXPERT KNOWLEDGE IN PROCESSING 2-D AND 3-D DATA, WITH AN EMPHASIS ON APPLICATIONS TO GEOSCIENCES AND BIOSCIENCES

by

ROBERTO ARAIZA, B.S., M.S.

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at El Paso

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

Department of Computer Science

THE UNIVERSITY OF TEXAS AT EL PASO

December 2007

Acknowledgements

First, I would like to thank Dr. Vladik Kreinovich for being my advisor. I would like to thank him for his time, his patience, and his willingness to teach me many things, not only in Computer Science or Math, but on virtually any other topic I may have been curious about. He has effectively become a role model for me: He is what I would like to become as a scientist, when I grow up :) I feel fortunate and grateful for having Dr. Kreinovich as my advisor, as he is as close to the ideal advisor as I could possibly want.

I would also like to thank Dr. Luc Longpré, Dr. Martine Ceberio, and Dr.Pavel Šolín for their willingness to serve on my committee. Their suggestions, comments, and additional guidance were very valuable to the completion of this work.

In addition, I would like to thank Dr. Pat Teller, Dr. Michela Taufer, and Dr. Ming-Ying Leung for supporting me as a Research Assistant while pursuing my degree. While working with them, I learned much on fields of Computer Science I wouldn't have been exposed to otherwise. Working with them was an invaluable experience, and I believe it truly helped me grow as a computer scientist.

También quiero agradecer a mi madre, Bertha Alicia Landeros Trevizo por ... todo. Su apoyo incondicional todos estos años es lo único que hizo posible que yo siguiera con mis estudios hasta ahora. No hay palabras suficientes para agradecerle todo lo que hace por mi, pero igual lo intentaré: ¡Gracias Ma! Esto es para usted.

NOTE:

This thesis was submitted to my Supervising Committee on November 12, 2007.

Abstract

Processing different types of data is one of the main applications of computers, the application for which computers have been originally designed. The more data we need to process, the more computing power we need and thus, the more important it is to use (and design) faster algorithms for data processing. Even processing 1-D data often is very time-consuming, but processing 2-D data (e.g., images) and 3-D data is where we really encounter the limits of the current computer hardware abilities.

Data processing algorithms sometimes produce results which contradict the expert knowledge – because this knowledge was not taken into account in the algorithm. For example, a mathematical solution to a seismic inverse problem may lead to un-physically large values of density inside the Earth.

At present, in such situations, researchers try to repeatedly modify ("hack") the process until the results produced by the algorithm agree with this expert knowledge. This process takes up a lot of expert time and – because of the need for numerous iterations – a lot of computer time.

To avoid this long, ad-hoc process, it is desirable to explicitly incorporate the expert knowledge into the algorithms, so that the results are always consistent with the expert's knowledge. Expert knowledge often comes in the form of bounds (i.e., intervals) on the actual values of the physical quantities. In this dissertation, we describe how this interval-related expert knowledge can be used in processing 2-D and 3-D data.

In data processing, one can distinguish between two types of situations: simpler situations when we directly measure the data that needs to be processed, and more complex situations when the data points can only be measured indirectly, i.e., when these points themselves need to be determined from the measurement results. In this dissertation, we consider both types of data processing. For the case of directly measured 2-D and 3-D data (e.g., images), one of the main problems usually is referencing these images. For the

case of indirectly measured data (e.g., for the case of an inverse problem), we need to *solve* this inverse problem, and we need to estimate how close the resulting solution is to the actual image. In this dissertation, we provide sample algorithms and examples showing how interval-related expert knowledge can help in solving the inverse problem and how interval techniques can help in determining how close the resulting solution is to the actual image. Most of the examples are from the geosciences, with some bioscience-related examples as well.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Need for Data Processing

In many real-life situations, there are quantities which are difficult (or even impossible) to measure directly: e.g., the amount of oil in an oil field or the angles between consequent atomic connections in a protein. Since we cannot measure the corresponding quantity *directly*, we can measure it *indirectly*: by measuring the values of easier-to-measure quantities $x_1, \ldots, x_n$ which are related to the desired quantity y by a known dependence.

This dependence can be *explicit*, when $y = f(x_1, \ldots, x_n)$ for a known function f, and there is a known algorithm which, given n real numbers $x_1, \ldots, x_n$, computes the value $f(x_1, \ldots, x_n)$ of this function with a reasonable accuracy. This algorithm can be as simple as an explicit computation of an analytical expression or as complex as solving a system of partial differential equations (PDEs).

In other practical situations, we only know an *implicit* dependence between x_i and y, i.e., a dependence of the form $F(x_1, \ldots, x_n, y) = 0$.

The resulting *indirect* measurement consists of the following:

- First, we measure the quantities $x_1, \ldots, x_n$.
- After that, we use the results $\widetilde{x}_i$ of directly measuring x_i (and the known dependence between x_i and y) to determine the value of the desired quantity y.

For example, in the case of an explicit dependence, we simply apply the function f to the measured values $\widetilde{x}_1, \ldots, \widetilde{x}_n$, and end up with an estimate $\widetilde{y} = f(\widetilde{x}_1, \ldots, \widetilde{x}_n)$ for the desired quantity y.

In computer terms, the computation that transforms the values $\widetilde{x}_1, \ldots, \widetilde{x}_n$ into an estimate $\widetilde{y}$ is called *data processing*. Data processing is one of the main applications of computers, the application for which computers have been originally designed.

1.2 Processing 2-D and 3-D Data: Computational Challenge

The more data we need to process, the more computing power we need and thus, the more important it is to use (and design) faster algorithms for data processing. Most algorithms for processing 2-D and 3-D data originated from similar 1-D algorithms. For example, Fourier transform was originally invented for 1-D data and later extended to 2-D and 3-D cases. Similarly, many ideas behind algorithms for solving (multi-D) partial differential equations (PDEs) originated in solving (1-D) ordinary differential equations (ODEs).

Usually, the transition from 1-D to 2-D or 3-D data means a drastic increase in the amount of data. In general, the more data we need to process, the more computing power we need. Thus, the same algorithm requires much more time to process 2-D or 3-D data than 1-D data. Even processing 1-D data is often very time-consuming. Processing 2-D data (e.g., images) and 3-D data (e.g., 3-D "images" describing the structure of the Earth or the structure of a biological object) is where we really encounter the limits of the current computer hardware abilities, where the latest record-breaking supercomputers need to be used.

It is therefore desirable to develop new techniques for processing 2-D and 3-D data.

1.3 Need for Expert Knowledge

The main objective of this dissertation is to describe how to use interval-related *expert knowledge* in processing 2-D and 3-D data. In order to better explain the need for expert knowledge, let us recall a typical manner in which mathematical and algorithmic techniques

are applied.

Most applications of mathematical techniques to practical problems follow the same paradigm.

First, we create a *model* for the desired real-life objects and processes. For example, Navier-Stokes equations are known to be a good model for the physical behavior and dynamics of liquids. A model needs to be *validated* to make sure that it adequately reflects the modeled phenomenon. It is worth mentioning, however, that even the best model is only an *approximation* to the real-life behavior. There is an inevitable *modeling error*; this error is small for good models, but it still needs to be taken into account.

Second, we use mathematical (numerical) methods to *solve* the equations which form this model. A numerical method needs to be *verified* to make sure that it provides an accurate solution to the mathematical model. Even the best numerical methods provide only an *approximation* to the desired solution; there is an inevitable *numerical (approximation) error*.

Because of these two types of errors, the result of data processing is generally different from the actual value of the corresponding physical quantity. So, within this paradigm, to decrease this difference, we must look for a more accurate model and/or for a more accurate numerical technique for solving this model.

To use a simplified example, if the model consists of Newton's equations of motion $\frac{d^2x}{dt^2} = \frac{F(x)}{m}$ and we know the mass m, the initial location $x(t_0)$ and velocity $\frac{dx}{dt}(t_0)$, and the dependence $F(x)$ of the force F on location x, then we can uniquely predict the location $x(T)$ of the particle at the desired future moment of time $T > t_0$. In real life, we may know the force $F(x)$ only approximately; this will constitute a modeling error. Also, the numerical integration method that we use to solve this equation has an approximation error.

In most real-life applications, this application paradigm works well, but in many other applications, the situation is more complex. Indeed, the traditional application paradigm implicitly assumes that the equations which comprise the model uniquely predict the values

of the desired quantity. In some practical applications, this is not always the case.

For example, one of the main objectives of *geophysics* is to determine the 3-D structure of the Earth. Different crustal materials usually have different density and, as a result, different velocity of sound v. Therefore, one way to determine the desired structure is to determine the velocity $v(\vec{x})$ at different 3-D points $\vec{x} = (x_1, x_2, x_3)$. To find these velocities, we set up explosions at different locations on the Earth surface, place sensors around these locations, and measure the time for the resulting seismic wave to propagate to the sensors. Based on these measured travel-times t_i and known locations of sensors, we then reconstruct the velocity model. The corresponding reconstruction problem is called the *seismic inverse problem*.

In the seismic inverse problem, it is usually assumed that each seismic wave follows the shortest path between the source and the sensor. This assumption can be viewed as a model of the seismic wave propagation. In terms of this model, the seismic inverse problem can be formally formulated as follows:

- We are given a number n (number of sensor readings).
- For each i from 1 to n, we know a spatial point S_i (location of the source), a spatial point R_i (location of the sensor), and the travel time t_i.
- We need to find a function $v(\vec{x})$ such that for each i,

$$\min_{\gamma: S_i \to R_i} \int_\gamma \frac{ds}{v(\vec{x})} = t_i,$$

  where the minimum is taken over all possible paths γ which start at the point S_i and end at the point R_i.

In reality, this model is only approximate because it assumes that the rays travel along a single path and ignores the effects of *diffraction*, when a wave "spreads out" beyond the path. Also, there are numerical errors caused by the fact that we usually use approximate methods to solve the corresponding system of equations. In most instances of the seismic inverse problem, errors of both types are usually very small.

However, in this problem, there is an additional source of error caused not so much by the *approximate* character of the model but rather by its *incompleteness*. Specifically, the travel-times only carry information about the locations through which the seismic waves pass. Many 3-D points are not covered by the corresponding ray paths. As a result, based only on the travel-times and on the shortest-path model, we cannot uniquely determine the velocity at these points.

The incompleteness is typical in numerical mathematics. Usually, it is resolved by making additional assumptions such as an assumption of smoothness (differentiability) – that the desired functions (such as our $v(\vec{x})$) smoothly depends on the coordinates. This smoothness assumption is behind the *regularization* techniques for solving the corresponding problems.

In the seismic inverse problem, it is known that the dependence $v(\vec{x})$ is, in reality, not smooth: there exist layers of different rocks, with an abrupt (discontinuous) transition between them. As a result, in general, we may have different layered solutions each of which is consistent with the same measurement results.

From the viewpoint of the existing mathematical formulation of the seismic inverse problem – via an incomplete model – all these drastically different solutions are all satisfactory. However, in practice, when a geophysicist expert looks at these solutions, he or she is normally able to select one (or a few) of these solutions which are consistent with the expert's understanding of geophysics – and dismiss other solutions as geophysically meaningless.

In such situations, it is desirable to use this expert information to "complete" the incomplete model. In the ideal world, we should be able to formulate the expert knowledge in a formalized way and thus, replace the original incomplete model with a (more) complete one – that combines the original equations with the formalized expert knowledge. In practice, the situation is much more "messy". For example, in the seismic inverse problem, the fact that we can have many drastically different solutions to each instance of this problems means that we can have many different algorithms which produce a solution

for each such instance – and in general, for the same instance, these algorithms lead to different solutions. A typical approach to incorporating the expert knowledge is to have experts select an algorithm which seems to be providing, in general, the best solutions, and to use thus selected algorithm to solve the problem.

For example, in the seismic inverse problem, Hole's code [89] is the algorithm that has been selected by the geophysics community and which is now most widely used to solve this problem.

The results of the selected algorithm are better not because they are in better accordance with the original (incomplete) model – all compared algorithms are in approximately the same good level of accordance. The selected algorithm is "better" because it better reflects the expert knowledge.

In this sense, the algorithm (method) becomes, in effect, a part of our model of the physical reality – the part which reflects expert knowledge.

In short, we select a method that is, *on average*, the best reflection of the expert knowledge, and we hope that this method provides a geophysically meaningful solution $v(\vec{x})$ in *all* (or at least in *most*) practical instances of the seismic inverse problem. Sometimes, the selected method does lead to a meaningful solution, but in many practical situations, the resulting solution is not in full agreement with the expert's intuition.

In some situations, a part of expert's knowledge can be explicitly formulated. For example, a geophysicist may know that the velocity at a certain depth must lie within certain reasonable bounds, e.g., between 5 and 8 km/s. At present, in such situations, researchers try to repeatedly modify ("hack") the process until the results produced by the algorithm agree with this expert knowledge. This process takes up a lot of expert time and – because of the need for numerous iterations – a lot of computer time. To avoid this long, ad-hoc process, it is desirable to explicitly incorporate the expert knowledge into the algorithms, so that the results are always consistent with the expert's knowledge.

Expert knowledge usually comes in the form of bounds (i.e., intervals) on the actual values of the physical quantities. It is therefore desirable to take this interval-related expert

knowledge into account when processing data.

1.4 Main Objective of the Dissertation

The main objective of the dissertation is to develop techniques for using such interval-related expert knowledge in processing 2-D and 3-D data.

1.5 Classification of the Related Problems

In data processing, one can distinguish between two types of situations:

- simpler situations when we directly measure the data $x_1, \ldots, x_n$ that needs to be processed, and
- more complex situations when the data points themselves can only be measured indirectly, i.e., when these points need to be determined from the measurement results.

In the dissertation, we provide algorithms and examples related to both types of data processing.

1.6 Simpler Problems

The simpler case is the case when we have directly measured 2-D and 3-D data. Some of these data sets come from an image, e.g., from a satellite or aerial photo. In such *remote sensing* situations, we have, in effect, a single "measurement" (in this case, the process of taking a photo) which produces all the data points. The corresponding intensities are usually measured with high accuracy. The only information that sometimes is lacking – and that we need to recover by processing the data – is the information about the exact spatial locations in which these intensities are measured.

In other cases, e.g., for gravity or magnetic data, we perform many measurements of the corresponding gravity or magnetic fields at different locations. In such situations, both the intensity and the location are measured with a reasonable accuracy. In computer science, the resulting 2-D and 3-D data are also usually called *images*.

As we have just mentioned, one of the main computational problems in processing remote sensor images is to *reference* the corresponding image (or images). Usually, when we have a new image that requires referencing, we already have geo-referenced previous images of the same area. So, a natural way to reference an image is to compare it with the previous images of the same area, and extract the spatial coordinates of the new points from this comparison.

This image referencing is one of the main problems that we handle in the dissertation. We describe the problem and our related ideas in Chapter 2.

For 2-D satellite images, the location of the image is determined modulo shift, rotation, and scaling. Thus, for such images, referencing two images means finding the shift, rotation, and scaling necessary to align one image with the other.

Referencing also is important for 3-D images. For example, in bioinformatics, it is important to find how a protein can dock to a ligand or another protein, i.e., to reference the corresponding 3-D images. This docking information is important because it can help in determining how medicine can help in preventing and curing diseases.

1.7 More Complex Problems

In more complex situations, we do not measure the image intensity directly, we need to compute the image values from the results of related measurements. An example of such a situation is a seismic inverse problem, where we need to reconstruct the values of the density and velocity in the Earth from the traveltimes of (natural and artificial) seismic signals.

In such situations, we face two main challenges:

- First, we need to *solve* the corresponding inverse problem.
- Second, we need to estimate how close the resulting solution is to the actual image.

In this dissertation, we provide algorithms (and examples) both for solving the inverse problem and for estimating how close the resulting solution is to the actual image. Most of the examples will be from the geosciences, with some bioinformatics-related examples as well. The corresponding formulations, ideas, and examples are described in Chapters 3 and 4.

Chapter 2

Image Referencing Using Fast Fourier Transform

In this dissertation, the simpler case of data processing we consider is the case when we have directly measured 2-D and 3-D data. This is the case with satellite images, aerial images, and many others; in computer science, the corresponding data sets are usually called images even when they do not represent a visual image. In such situations, the corresponding intensities are directly measured. The only information that sometimes is lacking – and that we need to recover by processing the data – is the information about the exact spatial locations in which these intensities are measured.

In other words, in such situations, one of the main computational problems is to *reference* the corresponding image (or images). Usually, when we have a new image that requires referencing, we already have referenced previous images of the same area. So, a natural way to reference an image is to compare it with the previous images of the same area, and extract the spatial coordinates of the new points from this comparison.

Image referencing is one of the main problems that we address in the dissertation.

2.1 Image Referencing: A Practical Problem

Formulation of the problem. In many areas of science and engineering, we have two images $I_1(\vec{x})$ and $I_2(\vec{x})$ (2-D or 3-D) which represent the same object but viewed from different angles and positions, in a possibly different scale.

In general, to describe the transition between different views, we must take into account

spatial shifts, rotations, scalings, and perspective distortions. Perspective distortions are extremely important in 2-D situations like processing security camera images or robotic vision. In these situations, it is important to compare the actual image of an object with a stored image of this object (and other objects), an image taken from an angle which is, in general, different from the current observation angle. Perspective distortion comes from the fact that 2-D images are obtained, in effect, by projecting a 3-D scene onto the 2-D image plane. As a result, for 3-D images, perspective distortion is not a problem.

As we have mentioned in Chapter 1, the more data points, the more challenging the corresponding image processing problem. We also mentioned that usually, 3-D images contain many more data points than 2-D images. So, the most computationally challenging case of image referencing is referencing 3-D images. In view of this fact, in this dissertation, we mainly concentrate on referencing 3-D images – and on the 2-D image referencing problems and algorithms which can be generalized to the 3-D case. Since in the 3-D case, the perspective distortion is not an issue, we therefore concentrate on the situations when the two referenced images $I_1(\vec{x})$ and $I_2(\vec{x})$ can be obtained from each other by an appropriate shift $\vec{x} \to \vec{x} + \vec{a}$, rotation $\vec{x} \to R\vec{x}$, and scaling $\vec{x} \to \lambda \cdot \vec{x}$ – without perspective distortion. In other words, we consider situations in which $I_2(\vec{x}) \approx I_1(\lambda \cdot R\vec{x} + \vec{a})$.

Such situations are not limited only to 3-D images. For example, satellite images are usually captured from such large distances that we do not need to consider perspective view. So, for satellite images, we can also safely assume that $I_2(\vec{x}) \approx I_1(\lambda \cdot R\vec{x} + \vec{a})$.

In many practical situations, we do not know the relative orientation of the two images. In such situations, it is desirable to *reference* these images, i.e., to find the rotation, the scaling, and the shift after which the images match as much as possible.

A similar problem occurs when we have the images of two different objects whose shapes should match. For example, we may have images of two bioactive molecules. We know that in vivo, these molecules interact because one of these molecules "docks" to the other one, i.e., gets into a position where their surfaces match. In such situations, it also is important to find the orientation and shift corresponding to this match.

Comment. Sometimes, the images also differ in lighting conditions, as a result of which we may have $I_2(\vec{x}) \approx C \cdot I_1(\lambda \cdot R\vec{x} + \vec{a})$ for some unknown factor C.

General image referencing problem vs. its particular cases. In the *general* problem, images may differ by shift, by orientation, and by scale. In such a general case, we need to find all three groups of parameters: shift $\vec{a}$, rotation R, and scaling λ.

In practice, sometimes we do not need to find all three of these parameters. For example, sometimes we know that the images already are similarly oriented and they are in the same scale, so all we need to do is to find the shift $\vec{a}$ between the two images.

In other practical problems, we know that the images are scaled right, but we need to find the shift $\vec{a}$ and the rotation R. For example, when a molecule docks into another molecule, it can shift and rotate, but it cannot seriously shrink or expand. Thus, in the protein docking problem, we must find the shift $\vec{a}$ and the rotation R – but not the scaling.

In short, in general, all three groups of parameters are unknown: the shift $\vec{a}$, the rotation R, and the scaling λ. In practice, in addition to this complex general image referencing problem, we sometimes face (somewhat) simpler particular cases of the general image referencing problem, cases in which we only need to determine some of these parameters.

How an image is represented. In order to formulate image referencing as an algorithmic problem, we need to describe how images are represented in a computer.

Usually, images are represented by the intensity values at different points on a rectangular grid. In particular, in the 2-D case, we have an $n \times n$ grid consisting of n^2 points. So, an image is represented by n^2 intensity values at different grid points. In the 3-D case, we have a 3-D $n \times n \times n$ grid consisting of n^3 points, so an image is represented by n^3 intensity values at different grid points. In general, an image is represented by n^d intensity values, where $d = 2$ or 3 is the dimension of the image.

2.2 First Case: Referencing of Similarly Oriented 2-D and 3-D Images – Existing Algorithms

Formulation of the practical problem: reminder.

- We have two images $I_1(\vec{x})$ and $I_2(\vec{x})$ (represented by their intensity values on a grid).
- We know that these images are obtained from each other by an (unknown) shift $\vec{a}$, i.e., that $I_2(\vec{x}) \approx I_1(\vec{x} + \vec{a})$.
- We want to find this shift.

Referencing in time $O(n^d)$. In some real-life problems, we can reference two similarly oriented images in the smallest possible time $O(n^d)$. One example is when we are referencing two images which have clear *landmarks*: e.g., points where the intensity has the largest possible value, which are useful in describing the structure of the image.

In such situations, to match the two images, it is sufficient to find, for each of the two images, the point $\vec{x}$ with the largest possible intensity value; this point can be found by searching over all n^d points $\vec{x}$ so it can be done in $O(n^d)$ steps. Once the corresponding landmark locations $\vec{d}_1$ and $\vec{d}_2$ are found, we can find the desired shift $\vec{a}$ as $\vec{a} = \vec{d}_2 - \vec{d}_1$.

For many important images, we cannot use this fast algorithm for image referencing, because there are no easy-to-find landmark locations – namely, for every special point in the image, there are several similar points in the same image. For example, in multi-cellular biological images, whether we have a point on the border between several cells, or a special point of a special structure within a cell, there are many different points with the similar values of intensity at different cells. Similarly, in a satellite image, if the maximum of the image intensity corresponds to, e.g., road intersection or a river, we may have many road intersections and several rivers within the same image.

In some such situations, however, we can still use a fast ($O(n^d)$) algorithm for image referencing. For example, in astronomical images, we often have an image of an object

surrounded by empty space. In this case, we can use the moments to find out the shift between the images. It is easy to see that when we apply a shift to an image, i.e., replace the original image $I_1(\vec{x})$ with a shifted image $I_2(\vec{x}) = I_1(\vec{x} + \vec{a})$, then the 0-th order moment (overall intensity) $M_i = \int I_i(\vec{x})\, d\vec{x}$ does not change ($M_2 = M_1$), while the 1-st order moments $M_{ij} = \int I_i(\vec{x}) \cdot x_j \cdot d\vec{x}$ change to $M_{2i} = M_{1i} - a_i \cdot M_1$. Thus, we can find the desired shift $\vec{a}$ by comparing the corresponding moments: $a_i = (M_{1i} - M_{2i})/M_1$. An integral is, in effect, a sum over all n^d grid points, so computing each of the integrals M_1 and M_{ij} requires $O(n^d)$ steps. Thus, we still need computation time $O(n^d)$.

This method has a natural geometric meaning: the ratio M_{ji}/M_1 is the i-th coordinate of the image's center of gravity. Thus, in effect, this algorithm consists of the following three steps:

- first, we compute the center of gravity $d_{1i} = M_{1i}/M_1$ of the image $I_1(\vec{x})$;
- then, we compute the center of gravity $d_{2i} = M_{2i}/M_2$ of the image $I_2(\vec{x})$;
- finally, we find the desired shift as the difference between the centers of gravity: $\vec{a} = \vec{d_1} - \vec{d_2}$.

Comment. This reformulation works also when the images are not only shifted, but also "re-scaled", i.e., $I_2(\vec{x}) \approx C \cdot I_1(\vec{x} + \vec{a})$ for some (unknown) constant $C > 0$, due, e.g., to different lighting conditions related to these two images.

Situations when $O(n^d)$ algorithms are not applicable. In many practical situations, none of the above $O(n^d)$ algorithms is applicable. For example, in satellite and multi-cellular images, not only we do not have landmark points, we also do not have empty space around an image. As a result, e.g., the shifted satellite image contains values which are not present in the original image. When both images cover the same reasonably homogeneous area, then the intensity is of about the same value throughout both images, and hence, the center of gravity of each image is close to the geometric center of the image. Thus, if we apply the

above center-of-gravity algorithm to detect the shift between the images, then, no matter what the actual shift is, the above algorithm will return (approximately) 0.

Straightforward approach: computational complexity. A natural idea is to look for a vector $\vec{a}$ for which the distance

$$e(\vec{a}) \stackrel{\text{def}}{=} \int (I_2(\vec{x}) - I_1(\vec{x} + \vec{a}))^2 \, d\vec{x} \tag{2.1}$$

attains the smallest possible value.

The minimized objective function is, in effect, the sum of the squares. Thus, the corresponding minimization problem is a particular case of the Least Squares Method (LSM). In practice, a standard way to solve a LSM problem is to equate the derivative of the corresponding quadratic function to 0; since we start with a quadratic function, the derivative is linear and so, we get a system of linear equations for finding the desired values. However, in our case, this approach cannot be easily applied. Indeed, differentiating the integral 2.1 with respect to the unknowns $\vec{a}$ means that we should differentiate the function $I_1(\vec{x} + \vec{a})$. However, the images $I_1(\vec{x})$ and $I_2(\vec{x})$ are, in general, not smooth (i.e., not everywhere differentiable). Often, an image contains an abrupt boundary: e.g., satellite images can have a boundary between the ocean and the land, between the glacier and the surrounding area, etc. So, we cannot use the usual LSM simplification to solve our minimization problem, we need to find other ways to solve this problem.

Let us start with a straightforward approach: we compute the value of the "scoring function" $e(\vec{a})$ for all computer representable shifts $\vec{a}$, and then find the shift for which the value $e(\vec{a})$ is the smallest. As we will see, this approach is not practical for real-life values n.

Each image on a 2-D $n \times n$ grid consists of n^2 intensity values at different grid points. Computing each integral requires time $O(n^2)$: for each pixel $\vec{x}$, we need one subtraction and one multiplication to compute the value $(I_2(\vec{x}) - I_1(\vec{x} + \vec{a}))^2$, and then we must add all these values to get the integral.

According to the straightforward approach, we must compute the value of the L^2-norm for all possible shifts $\vec{a}$. For an $n \times n$ grid, it is reasonable to consider n^2 possible shifts. Computing each integral requires time $O(n^2)$, so overall, we need time $O(n^2) \cdot O(n^2) = O(n^4)$.

For many real-life images, the size n is approximately 10^3, so $n^4 \approx 10^{12}$ computational steps require a large amount of time. It is therefore desirable to find faster algorithms for image referencing.

It is known that many signal and image processing techniques can be made faster if we use Fast Fourier Transform (FFT); see, e.g., [47]. Fourier transform $F(\vec{\omega})$ of an image $I(\vec{x})$ is defined as

$$F(\vec{\omega}) = \frac{1}{(\sqrt{2\pi})^d} \cdot \int I(\vec{x}) \cdot \exp(-\mathrm{i} \cdot \omega \cdot \vec{x})\, d\vec{x},$$

where $d = 2, 3$ is the dimension of the space, $\mathrm{i} \stackrel{\text{def}}{=} \sqrt{-1}$, and $\vec{\omega} \cdot \vec{x} \stackrel{\text{def}}{=} \omega_1 \cdot x_1 + \omega_2 \cdot x_2 + \ldots$ is a scalar (dot) product of the two vectors.

FFT is an algorithm for computing the Fourier transform of a given image. The application of FFT to an image of size N requires $N \cdot \log(N)$ steps, so for an image of size $N = n^d$, we need $O(n^d \cdot \log(n))$ steps.

Comment. The standard FFT algorithm requires that the number n of pixels along each image dimension is a power of 2, i.e., $n = 2^k$ for some non-negative integer k. Images for which n is not a power of 2 can be "padded" with zeros to the nearest 2^k size. For example, images of size 1,000×1,000 are "padded" to size 1,024×1,024.

Using Fast Fourier Transform for shift detection. For a shifted image $I_2(\vec{x}) = I_1(\vec{x} + \vec{a})$, the Fourier transform has the form $F_2(\vec{\omega}) = \exp(\mathrm{i} \cdot \vec{\omega} \cdot \vec{a}) \cdot F_1(\vec{\omega})$. Thus, in the presence of measurement noise and inaccuracy, when we have $I_2(\vec{x}) \approx I_1(\vec{x} + \vec{a})$, we have $F_2(\vec{\omega}) \approx r(\vec{\omega}) \cdot F_1(\vec{\omega})$, where $r(\vec{\omega}) \approx \exp(\mathrm{i} \cdot \vec{\omega} \cdot \vec{a})$.

The complex number $\exp(\mathrm{i} \cdot \vec{\omega} \cdot \vec{a})$ has magnitude 1; so, it is reasonable, for every spatial frequency $\vec{\omega}$, to find the value $r(\vec{\omega})$ for which the approximation error $|F_2(\vec{\omega}) - r(\vec{\omega}) \cdot F_1(\vec{\omega})|^2$

is the smallest possible under the constraint that $|r(\vec{\omega})| = 1$.

By using the Lagrange multiplier method, we can show that the solution to this constrained optimization problem is given by

$$r(\vec{\omega}) = \frac{F_1(\vec{\omega}) \cdot F_2^*(\vec{\omega})}{|F_1(\vec{\omega}) \cdot F_2(\vec{\omega})|}. \tag{2.2}$$

In the ideal case, $r(\vec{\omega})$ is a sinusoidal wave, and its inverse Fourier transform is an impulse function $\delta(\vec{x}-\vec{a})$, i.e., a function which only is non-zero for $\vec{x} = \vec{a}$. It is therefore reasonable to find the shift $\vec{a}$ by applying the inverse FFT to the above function $r(\vec{\omega})$ and to find the shift as the point at which this inverse FFT attains the largest possible value.

We thus arrive at the following algorithm.

Resulting algorithm. We are given two images $I_1(\vec{x})$ and $I_2(\vec{x})$; we must find the shift $\vec{a}$ between these images. For that, we can do the following:

- first, we apply FFT to both images $I_1(\vec{x})$ and $I_2(\vec{x})$, and compute the Fourier transforms $F_1(\vec{\omega})$ and $F_2(\vec{\omega})$;
- then, for each frequency $\vec{\omega}$, we compute the ratio (2.2);
- after that, we apply the inverse FFT to the resulting function $r(\vec{\omega})$ and obtain a new function $P(\vec{x})$.

Finally, we find the shift $\vec{a}$ as the value at which the function $P(\vec{x})$ attains its largest possible value.

Comment. This algorithm was first proposed and implemented in [122] (without the least squares justification); for 2-D images, it requires time $O(n^2 \cdot \log(n))$. This same algorithm can be used for 3-D images, it then requires time $O(n^3 \cdot \log(n))$.

2.3 Referencing of General 2-D and 3-D Images: Existing Algorithms

Formulation of the problem. We have two d-D images $I_1(\vec{x})$ and $I_2(\vec{x})$; we must find the shift $\vec{a}$ and the rotation R for which $I_2(\vec{x}) \approx I_1(R\vec{x} + \vec{a})$.

Referencing in time $O(n^d)$. To find the shift between the two images, it is sufficient to find one landmark point. To find both shift and rotation, we can use two different landmarks:

- we can use the first landmark to find the shift between the two images, and then
- we can compare the location of the second landmark in both images, and thus find the corresponding rotation.

Finding the landmarks in the image requires time $O(n^d)$.

If there are no easily detectable landmarks, but the images are objects surrounded by an empty space, then we can use the moments method to determine both shift and rotation. The values of the 0-th and 1-st order moment M_i and M_{ij} could only determine the shifts. So, to find the rotation, we must also use the second order moments $M_{ijk} = \int I_i(\vec{x}) \cdot x_j \cdot x_k \, d\vec{x}$.

To compare the second order moments (also called moments of inertia) of the two images, it is necessary to first shift both images to the same point of origin, e.g., to the center of gravity. As a result, we replace the original value M_{ijk} with the new value $M_{ijk} - d_j \cdot d_k \cdot M_i$, where $d_j = M_{ij}/M_i$ is the j-th coordinates of the center of gravity.

Second order moments of each image form a symmetric non-negative definite matrix. This matrix is 2×2 in the 2-D case and 3×3 in the 3-D case. Since this matrix is symmetric, it has orthogonal eigenvectors. When the image rotates, these eigenvectors rotate as well. So, we can find the rotation angle by comparing the orientations of the eigenvectors. Since the sizes of the moments-of-inertia matrices are small (2×2 or 3×3 for 3-D), these eigenvectors can be computed very fast.

Using Fast Fourier Transform for rotation detection: 2-D case. In general, the images may not have clear landmarks and they may not be surrounded by an empty space, so the above $O(n^d)$ methods may not be applicable.

If the two images $I_1(\vec{x})$ and $I_2(\vec{x})$ differ not only by shift but also by a rotation R and a scaling λ, the absolute values $M_1(\vec{\omega})$ and $M_2(\vec{\omega})$ of their Fourier transforms $F_1(\vec{\omega})$ and $F_2(\vec{\omega})$ differ from each other only by the corresponding rotation and scaling.

In the 2-D case, rotation can be reduced to a shift if we use polar coordinates (r, θ). Indeed, in polar coordinates, rotation by an angle θ_0 is described by a shift-like formula $\theta \to \theta + \theta_0$. So, in polar coordinates, rotation is described by a shift. So, if we use the above FFT-based algorithm, we can find the shift R, and thus, determine the rotation angle R.

We can now apply the reconstructed rotation to one of the images, e.g., to the first image, $I(x, y)$, to obtain a new image, $\hat{I}(x, y)$. Since we rotated one of the images, the images are already aligned in terms of rotation, and the only difference between them is in an (unknown) shift. So, we can apply the above shift-detection algorithm for determining the shift.

This idea was first proposed in [182]; for 2-D images, it requires time $O(n^2 \cdot \log(n))$.

Limitations. The above idea only works for 2-D images. This idea cannot be directly applied to speed up referencing of 3-D images.

2.4 Remaining Problem

Summarizing: referencing two images, i.e., finding a shift and a rotation that match the two given images, is an important practical problem. In principle, it is always possible to solve this problem by trying all possible values of shift and rotation; however, this exhaustive search requires an un-realistically huge amount of computation time.

This amount of computations is large already for 2-D images, and it is even larger for 3-D images. There are two reasons for this drastic increase in computational complexity.

First, a 3-D image of linear size n contains n^3 different values of intensity – much more than n^2 values that form a 2-D image of the same linear size. Processing more data points requires more computation time.

Second, to find the right match between two 3-D images, we need more tests than for matching 2-D images. Indeed, e.g., to find the appropriate shift between 2-D images, we need, in the worst case, to test n^2 possible shifts. For the 3-D case, we need $n^3 \gg n^2$ possible shifts.

The difference is even larger if we take rotations into account. Indeed, to describe a rotation around a point in a 2-D space, it is sufficient to use a single parameter – the rotation angle. To describe a generic rotation in the 3-D case, we need 3 parameters. Thus, to find a rotation by the exhaustive search, we need $\approx n$ tests in the 2-D case, but $\approx n^3 \gg n$ tests in the 3-D case.

In the 2-D case, it is possible to reduce the problem of finding both rotation and shift to two simpler problems of finding shift. However, this reduction does not work in the 3-D case – the case when the computation time is the largest, and thus, there is the greatest need to speed up.

2.5 New Algorithms

Main idea. Our main idea is to combine the techniques from the existing efficient image referencing algorithms: the use of moments and the use of Fourier transforms.

Preliminary step: the general case of shift and rotation can be reduced to the case when the images only differ by rotation. In general, referencing the two images means finding the values of the shift and rotation (and sometimes also scaling) that transform the first image into the second one. We have mentioned that the computational complexity of the problem comes from the fact that we need a reasonably large number of parameters to describe possible combinations of shift and rotation.

To simplify this problem, let us reduce this problem to the case when the images differ only by rotation. This reduction was, in effect, explicitly described when we covered the existing 2-D FFT-based image referencing techniques. Specifically, this reduction is based on the fact that the absolute value $M(\vec{\omega}) = |F(\vec{\omega})|$ of the Fourier transform $F(\vec{\omega})$ does not change under shift. Thus, if the two images $I_1(\vec{x})$ and $I_2(\vec{x})$ differ not only by shift but also by a rotation R, the absolute values $M_1(\vec{\omega})$ and $M_2(\vec{\omega})$ of their Fourier transforms $F_1(\vec{\omega})$ and $F_2(\vec{\omega})$ differ from each other only by the corresponding rotation.

In view of this fact, in the following text, we will start with the absolute values $M_1(\vec{\omega})$ and $M_2(\vec{\omega})$, and we will try to find the rotation which transforms these absolute values into one another.

Once we know the rotation R, we can align the images – e.g., by rotating the Fourier transform $F_1(\vec{\omega})$ of the image $I_1(\vec{x})$ to $F_1(R\vec{\omega})$. Now, the images $I_1(R\vec{x})$ and $I_2(\vec{x})$ differ only by shift, so we can use the FFT-based $O(n^d \cdot \log(n))$ time method to find this shift.

First possibility: method of moments in the Fourier space. In some practical situations, the corresponding magnitudes of Fourier transforms are rapidly decreasing as the spatial frequency ω increases; see, e.g., [188, 189]. In effect, this means that the corresponding "images" in the Fourier space are surrounded by an empty space.

For this situation, we already know how to find the corresponding rotation: we can use the method of moments. Thus, we arrive at the following new algorithm.

First new algorithm. In situations when the magnitude of the Fourier transform decreases with frequency, we can use the moments method to find the appropriate rotation. Namely, based on the function $M_1(\vec{\omega})$, we compute the second moments

$$M_{1ij} \stackrel{\text{def}}{=} \int M_1(\vec{\omega}) \cdot \omega_i \cdot \omega_j \, d\vec{\omega};$$

similarly, based on the function $M_2(\vec{\omega})$, we compute the second moments

$$M_{2ij} = \int M_2(\vec{\omega}) \cdot \omega_i \cdot \omega_j \, d\vec{\omega}.$$

For each of these matrices, we compute the eigenvectors; by comparing the orientations of these eigenvectors, we can then find the desired rotation R in time $O(n^d)$.

Comment. Images are usually given on a 2-D or 3-D grid, in the shape of a box. When we rotate the images an transform them back into a box shape, the corner information disappears. The only values which are preserved after all rotations are the values inside a circle (sphere) subscribed into the box. Since the corner values cannot match anyway, it is reasonable, before computing the moments, to only consider values $M(\vec{\omega})$ inside this circle (sphere).

First new algorithm: computational complexity. Overall, in the d-D case, we thus need $O(n^d \cdot \log(n))$ time to compute the original Fourier transforms, $O(n^d)$ time to find the rotation, and then $O(n^d \cdot \log(n))$ time to find the shift – to the total of

$$O(n^d \cdot \log(n)) + O(n^d) + O(n^d \cdot \log(n)) =$$

$$O(n^d \cdot \log(n))$$

computational steps.

First new algorithm: analysis of applicability. Our first new algorithm is applicable when the magnitude of the Fourier transform decreases with frequency. How can we reformulate this formal criterion in terms of the original image?

One possibility for such a reformulation comes from the known relations between functions and their Fourier transforms.

First, it is known that the L^2-norm $\int |f(\vec{x})|^2 \, d\vec{x}$ of a function $f(x)$ is equal to the L^2-norm $\int |F(\vec{\omega})|^2 \, d\omega$ of its Fourier transform $F(\vec{\omega})$.

Second, it is known that if $F(\omega)$ is a Fourier transform of a 1-D function $f(x)$, then for the derivative $f'(x)$, the Fourier transform is equal to $(\mathrm{i} \cdot \omega) \cdot F(\omega)$. Similarly, for the second derivative $f''(x)$, the Fourier transform is equal to $(\mathrm{i} \cdot \omega)^2 \cdot F(\omega)$, etc. So, if a function $f(x)$ is smooth, i.e., if, e.g., the integral $\int |f'(x)|^2 \, dx$ is reasonably small, this means that the equal

integral $\int \omega^2 \cdot |F(\omega)|^2\, d\omega$ is also small. Similarly, if a function $f(x)$ is twice differentiable, i.e., if, the integral $\int |f''(x)|^2\, dx$ is reasonably small, this means that the equal integral $\int \omega^4 \cdot |F(\omega)|^2\, d\omega$ is also small. For this integral to be small, the values $|F(\omega)|$ must rapidly decrease with $|\omega|$. This is true for 2-D and 3-D images as well.

So, we conclude that if the original image is smooth, then the magnitude of its Fourier transform is rapidly decreasing as the spatial frequency ω increases.

In practice, images are often smooth, and for these images, we have indeed seen that the magnitudes of their Fourier transforms indeed decrease with ω; see, e.g., [188, 189].

First new algorithm: limitations. In some practical situations, images are not smooth. In such situations, the magnitudes of their Fourier transforms do not decrease with the frequency and thus, we cannot use the above algorithm. How can we determine shift and rotation of such images?

Towards the second new algorithm. An image usually contains a smooth part and abrupt non-smooth parts. Non-smooth parts may include point-like objects – e.g., stars in an astronomical image or molecules in a biomolecular image. Some images are mainly smooth, some other images (like the ones we just mentioned) consist mainly of abrupt transitions.

For referencing smooth images, we can use the first new algorithm. Let us see what we can do for referencing images consisting of non-smooth parts.

If we have one such non-smooth part – or a few – then we have an image surrounded by an empty space, and we can use the moments method for referencing – and there is no need for Fourier transforms at all. This happens, e.g., when we have two images of the same molecule.

The problem occurs when we have an image which consists of numerous abrupt objects, objects which are not limited to a region within the image. In this case, we cannot directly apply the moments method to the original images.

Instead, we can use the above-described relation between properties of a function and properties of its Fourier transform. Namely, we can use the fact that for a smooth image, its Fourier transform is located in the vicinity of 0 – and, correspondingly, that vice versa, for an image which is located in the vicinity of 0, its Fourier transform is smooth. This "reverse" conclusion can be justified directly: the Fourier transform is defined as $F(\vec{\omega}) = \int I(\vec{x}) \cdot \exp(-\mathrm{i} \cdot \vec{\omega} \cdot \vec{x})\, d\vec{x}$. We can differentiate the expression for $F(\vec{\omega})$ relative to ω and get an expression for the derivative in terms of integrals containing $I(\vec{x})$. If the image $I(\vec{x})$ is mainly concentrated within several reasonably small areas, the corresponding integrals are small, and thus, the derivatives of $F(\vec{\omega})$ are also small – i.e., the Fourier transform is indeed smooth.

Let us use this fact in referencing. Let us consider images $I_1(\vec{x})$ and $I_2(\vec{x})$ which differ by shift and rotation. In this case, their Fourier transforms $F_1(\vec{\omega})$ and $F_2(\vec{\omega})$ are smooth.

We have already mentioned that if the two images $I(\vec{x})$ and $I'(\vec{x})$ differ only by shift, then the absolute values of their Fourier transforms coincide: $|F(\vec{\omega})| = |F'(\vec{\omega})|$. Thus, the absolute values $|F_1(\vec{\omega})|$ and $|F_2(\vec{\omega})|$ of the Fourier transforms of the original images differ only by rotation.

We already know the advantages of smoothness, so it would be nice to get smooth images which differ only by rotation. The Fourier transforms are smooth, but their absolute values are not necessarily smooth: indeed, the absolute value is not a smooth procedure because it is not differentiable at 0. To get smooth images, instead of the absolute value $|F(\vec{\omega})|$, it is reasonable to consider the square of the absolute values, i.e., the image $|F(\vec{\omega})|^2 = F(\vec{\omega}) \cdot F^*(\vec{\omega})$, where z^* denotes a complex conjugate of a complex number. The operation $z \to z \cdot z^* = |z|^2$ is smooth and thus, for smooth Fourier transforms $F_i(\vec{\omega})$, the functions $|F_i(\vec{\omega})|^2$ are also smooth.

Since these functions are smooth, their inverse Fourier transforms

$$J_i(\vec{x}) \stackrel{\mathrm{def}}{=} FFT^{-1}(|F_i(\vec{\omega})|^2)$$

rapidly decrease with $|\vec{x}|$ – in other words, they are mainly located in the small vicinity of

0. These new images $J_1(\vec{x})$ and $J_2(\vec{x})$ also differ from each other by (the same) rotation. Since these images are mainly located at 0, we can apply the original moments method to find this rotation – which will be exactly the rotation between the original images $I_1(\vec{x})$ and $I_2(\vec{x})$.

Thus, we arrive at the following algorithm.

Second new algorithm: a description. In situations when the image mainly consists of several abrupt components, we can use the moments method to find the appropriate rotation. Namely, after computing the absolute values $M_1(\vec{\omega})$ and $M_2(\vec{\omega})$ of the original images, we compute the inverse Fourier transforms $J_1(\vec{x}) = FFT^{-1}(M_1^2(\vec{\omega}))$ and $J_2(\vec{x}) = FFT^{-1}(M_2^2(\vec{\omega}))$ of their squares. Based on these new functions $J_1(\vec{x})$ and $J_2(\vec{x})$, we compute the second moments $M_{1ij} \stackrel{\text{def}}{=} \int J_1(\vec{x}) \cdot x_i \cdot x_j \, d\vec{x}$ and $M_{2ij} \stackrel{\text{def}}{=} \int J_2(\vec{x}) \cdot x_i \cdot x_j \, d\vec{x}$. For each of these matrices, we compute the eigenvectors; by comparing the orientations of these eigenvectors, we can then find the desired rotation R in time $O(n^d)$.

Second new algorithm: computational complexity. Overall, in the d-D case, we thus need $O(n^d \cdot \log(n))$ time to compute the original Fourier transforms, $O(n^d \cdot \log(n))$ time to apply the inverse Fourier transform and compute the auxiliary functions $J_1(\vec{x})$ and $J_2(\vec{x})$, $O(n^d)$ time to find the rotation, and then $O(n^d \cdot \log(n))$ time to find the shift – to the total of

$$O(n^d \cdot \log(n)) + O(n^d \cdot \log(n)) + O(n^d) + O(n^d \cdot \log(n)) =$$

$$O(n^d \cdot \log(n))$$

computational steps.

Two new algorithms: analysis of applicability. Our first new algorithm is applicable when the image is mainly smooth; our second algorithm is applicable when the image mainly consists of several abrupt components.

In some practical cases, we have images which contain smooth parts and non-smooth parts, none of which is prevailing. In this situation, we cannot apply either of our algorithms to reference the original images (and we did encounter such examples when we tested our algorithms on different images). In this situation, a natural recommendation is to look for *parts* of the original images which are either mainly smooth or mainly abrupt, and to reference these parts.

2.6 Testing of the New Algorithms

2.6.1 Testing on Simple Artificial Images

Before we describe the results of our testing, let us explain why we selected the specific test images.

Selection of test images: linear combination of Gaussian components. We first tested our algorithms on simple 3-D examples, for which we could derive analytical expressions for all intermediate functions. These intermediate functions include Fourier transforms of the original image, so we must make sure that we select simple functions for which we have analytical expressions for their Fourier transform.

Such analytical expressions are known for *Gaussian functions*, i.e., expressions similar to the probability density function of a normal (Gaussian) distribution. A Gaussian function with amplitude A and width σ centered at a point $\vec{a}$ is defined as

$$G(\vec{x}) = A \cdot \exp\left(\frac{(\vec{x}-\vec{a})^2}{2\sigma^2}\right).$$

For this Gaussian function, the Fourier transform $G_F(\vec{\omega})$ has a known analytical expression:

$$G_F(\vec{\omega}) = A \cdot \exp(\mathrm{i} \cdot \vec{\omega} \cdot \vec{a}) \cdot \exp\left(\frac{(\vec{\omega})^2}{2\sigma^{-2}}\right).$$

The absolute value of this Fourier transform is again a Gaussian function.

Since Fourier transform is a linear operation, the Fourier transform of a linear combination of several Gaussian functions can also be described by an explicit analytical expression – the corresponding linear combination of Fourier transforms of these Gaussian functions.

In view of this fact, in our tests, we used linear combinations of Gaussian functions; these images are described in detail in the following text.

If all these Gaussian functions are narrow, then we can simply use the moments method to find shift and rotation, and there is no need for our more time-consuming algorithms. Thus, to test our algorithms on examples on which the simple moments method may not work, we added a "spread" component to the point-like Gaussian functions. In the limit, when the width tends to ∞, we get a constant function $G_{\lim}(\vec{x}) = A$. For simplicity, we added such a constant term to the linear combination of narrow Gaussian functions.

Analytical formulas for the auxiliary image $J(\vec{x})$ (the inverse Fourier transform of $|F(\vec{\omega})|^2$). In the second new algorithm, we compute the Fourier transform $J(\vec{x})$ of the expression $|F(\vec{\omega})|^2 = F(\vec{\omega}) \cdot F^*(\vec{\omega})$. Let us show that for simple images, we can derive a simplified expression for this auxiliary image $J(\vec{x})$.

For that, we will find the inverse Fourier transforms of the images $F(\vec{\omega})$ and $F^*(\vec{\omega})$ and then combine these inverse Fourier transforms to get the desired inverse Fourier transform $J(\vec{x})$ of the product $F(\vec{\omega}) \cdot F^*(\vec{\omega})$.

By definition, $F(\vec{\omega})$ is the Fourier transform of the original image $I(\vec{x})$. Thus, the inverse Fourier transform of $F(\vec{\omega})$ is exactly the original image $I(\vec{x})$.

Let us now find the inverse Fourier transform of the complex conjugate function $F^*(\vec{\omega})$. Since

$$F(\vec{\omega}) = \int I(\vec{x}) \cdot \exp(-\mathrm{i} \cdot \vec{\omega} \cdot \vec{x})\, dx,$$

its complex conjugate $F^*(\vec{\omega})$ is equal to

$$F^*(\vec{\omega}) = \int I(\vec{x}) \cdot \exp(\mathrm{i} \cdot \vec{\omega} \cdot \vec{x})\, dx.$$

We want to find the inverse Fourier transform of the function $F^*(\vec{\omega})$. In other words, we want to represent $F^*(\vec{\omega})$ as a Fourier transform of some function. The above expression is

similar to the expression for the Fourier transform, with the only difference that we have $\mathrm{i} \cdot \vec{\omega} \cdot \vec{x}$ instead of $-\mathrm{i} \cdot \vec{\omega} \cdot \vec{x}$. To take care of this difference, let us introduce new variables $\vec{y} = -\vec{x}$ for which $\vec{x} = -\vec{y}$. In terms of $\vec{y}$, the above formula takes the form

$$F^*(\vec{\omega}) = \int I(-\vec{y}) \cdot \exp(-\mathrm{i} \cdot \vec{\omega} \cdot \vec{y})\, dy.$$

So, the complex conjugate function $F^*(\vec{\omega})$ is equal to the Fourier transform of the function $I(-\vec{x})$ and thus, the inverse Fourier transform of $F^*(\vec{\omega})$ is equal to $I(-\vec{x})$.

Let us now combine the inverse Fourier transforms of $F(\vec{\omega})$ and $F^*(\vec{\omega})$ into the inverse Fourier transform for their product. It is known that, in general, the inverse Fourier transform $J(\vec{x})$ of the product $F_1(\vec{\omega}) \cdot F_2(\vec{\omega})$ is equal to the convolution of the corresponding inverse Fourier transforms $I_1(\vec{x})$ and $I_2(\vec{x})$, i.e.,

$$J(\vec{x}) = \int I_1(\vec{y}) \cdot I_2(\vec{x} - \vec{y})\, d\vec{y}.$$

Thus, the inverse Fourier transform $J(\vec{x})$ is equal to the convolution of the functions $I(\vec{x})$ and $I(-\vec{x})$, i.e., to

$$J(\vec{x}) = \int I(\vec{y}) \cdot I(\vec{y} - \vec{x})\, d\vec{y}.$$

We consider images consisting of a few point-like components, for which $I(\vec{x})$ is different from 0 only in small vicinities of a few points – centers of the corresponding Gaussian components. In this case, according to the above formula for $J(\vec{x})$, the value $J(\vec{x})$ is different from 0 if and only if there exist values $\vec{y}$ and $\vec{z} \stackrel{\text{def}}{=} \vec{y} - \vec{x}$ at which the original image I is different from 0. In other words, $J(\vec{x})$ is different from 0 only when for some point $\vec{y}$ which is close to one of the centers, the difference $\vec{z} = \vec{y} - \vec{x}$ is close to some other component center. The value $\vec{x}$ is equal to the difference between these two center locations:

$$\vec{x} = \vec{y} - (\vec{y} - \vec{x}) = \vec{y} - \vec{z}.$$

So, to find all the points where this auxiliary image is different from 0, it is sufficient to take the differences between all the component centers.

Simplest case with non-trivial referencing problem: 3 Gaussian components. We would like to make our test images as simple as possible. In particular, we would like to use test images which contain as few components as possible.

Let us start with considering the case when we have only one component, i.e., the case when the image is described by a Gaussian function: $I_1(\vec{x}) = G(\vec{x})$. The expression for the Gaussian function $G(\vec{x})$ depends only on the distance $\|\vec{x} - \vec{a}\|$ between a given point $\vec{x}$ and the center $\vec{a}$ of this function. This distance does not change if we simply rotate the image around the center $\vec{a}$. Thus, if we rotate the image $I_1(\vec{x})$ around the center $\vec{a}$, the image will not change at all: $I_2(\vec{x}) = I_1(\vec{x})$. Since the image does not change under rotation, we cannot determine the rotation from the images $I_1(\vec{x})$ and $I_2(\vec{x})$.

If we rotate around any other point, we will encounter the same problem – except for a possible shift. Thus, for images consisting of a single Gaussian function, we do not have a non-trivial referencing problem.

Similarly, if we consider an image consisting of 2 Gaussian functions, then this image does not change if we rotate it around the axis connecting the centers of these Gaussian functions. So here too, based on simply knowing the two images $I_1(\vec{x})$ and $I_2(\vec{x}) = I_1(\vec{x})$, we cannot tell whether there was no rotation involved at all or there was actually a rotation around this axis. Thus, for images consisting of two Gaussian functions, we also do not have a non-trivial referencing problem.

Thus, the simplest case when we do have a non-trivial referencing problem is the case of images which consist of 3 Gaussian functions.

Specific selection of the first test image. As a first test image $I_1(x)$, we take the sum of three Gaussian functions with the same amplitude $A = 1$, the same width $\sigma = 0.5$, and with centers at the points $(n/2, n/2, n/2)$, $(3 \cdot n/4, n/2, n/2)$, and $(n/2, 3 \cdot n/4, n/2)$,

with $n = 64$. In other words, we consider the following image:

$$\begin{aligned} I_1(x,y,z) &= \exp([(x-(n/2))^2 + (y-(n/2))^2 + (z-(n/2))^2]/[2\cdot\sigma^2]) + \\ &\quad \exp([(x-(3\cdot n/4))^2 + (y-(n/2))^2 + (z-(n/2))^2]/[2\cdot\sigma^2]) + \\ &\quad \exp([(x-(n/2))^2 + (y-(3\cdot n/4))^2 + (z-(n/2))^2]/[2\cdot\sigma^2]), \end{aligned} \tag{2.3}$$

Figure 2.1 shows plane $z = 32$ of $I_1(x, y, z)$.

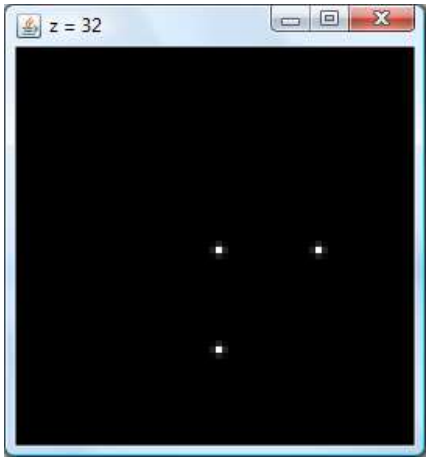


Figure 2.1: The first test image: a simple artificial image with the smallest number of Gaussian components.

Selection of shifts and rotations in testing experiments. To check how different algorithms work on this image, we need to shift and rotate it, resulting in a new image $I_2(\vec{x})$, and then reconstruct the shift and rotation from the images $I_1(\vec{x})$ and $I_2(\vec{x})$. In our tests, we used a shift by a vector $\vec{s} = (5, -4, 7)$; we tried other shifts, and the results are very similar.

A general rotation in a 3-D space can be described as a composition of three rotations: a rotation around the z axis, followed by a rotation around the y axis, and a rotation around the x axis. In the following test, we will denote the rotation angle of a rotation around the

x axis by α, the rotation angle of a rotation around the y axis by β, and the rotation angle of a rotation around the z axis by γ.

In contrast to shift, for which the reconstruction accuracy does not depend on the value of the shift, the accuracy with which we can reconstruct a rotation depends on this rotation. Thus, instead of selecting a single rotation, we tried several rotations. Specifically, for each of the three angles α, β, and γ, we tried 5 different values 0, 15, 30, 45, and 60 degrees. Overall, we have $5 \times 5 \times 5 = 125$ combinations; one of these combinations, however, is a trivial one $(\alpha, \beta, \gamma) = (0, 0, 0)$ for which there is no rotation at all. So, we have 124 non-trivial combinations.

Gauging the reconstruction quality. To test an algorithm, we run it on all 124 possible rotations (α, β, γ). Specifically, for each of these rotations, we apply this rotation to the image $I_1(\vec{x})$ producing the rotated image $I_2(\vec{x})$, and then apply the tested algorithm to the images $I_1(\vec{x})$ and $I_2(\vec{x})$. The rotation angles $(\alpha', \beta', \gamma')$ produced by the algorithm are then compared to the original rotation angles.

To gauge the quality of an algorithm, we compute the smallest ("best") and the largest ("worst") values of the corresponding 124 reconstruction errors

$$\max(|\alpha - \alpha'|, |\beta - \beta'|, |\gamma - \gamma'|).$$

Testing the moments of inertia algorithm on the first test image. Since the images $I_1(\vec{x})$ and $I_2(\vec{x})$ are, in effect, surrounded by empty space, even computing the moments of inertia of the images should work, with no need to use the absolute values of the Fourier transform. This is indeed the case; results are summarized in Table 2.1. In the worst case, we obtain an error of 1.78 degrees. This is reasonable; since we are distributing the possible range of 180 degrees (from -90 to 90) over 64 pixels, we have, within a pixel, an error of $180/64 \approx 2.8$ degrees.

Table 2.1: The results of testing the moments method on the first test image

Angle	Worst Run	Best Run
α	45.00°	15.00°
α'	44.93°	14.99°
β	30.00°	0.00°
β'	29.98°	−0.02°
γ	30.00°	60.00°
γ'	28.22°	60.01°
error	1.78°	0.02°

Testing the first new algorithm on the first test image. The image $I_1(\vec{x})$ consists of a few abrupt components. The first new algorithm (moments method in Fourier space) only works for smooth images. It is therefore reasonable to expect that the first new algorithm will not work on this image $I_1(\vec{x})$. This is indeed the case, as shown in Table 2.2.

Table 2.2: The results of testing the first new algorithm on the first test image

Angle	Worst Run	Best Run
α	60.00°	30.00°
α'	72.52°	33.09°
β	60.00°	30.00°
β'	−89.36°	26.64°
γ	0.00°	60.00°
γ'	−53.60°	58.62°
error	149.36°	3.36°

Testing the second new algorithm on the first test image. The case of an image consisting of a few abrupt components is exactly the case for which the second new algorithm was designed. An indeed, as shown in Table 2.3, the results of applying this method are very good: as good as for the original moments method.

Table 2.3: The results of testing the second new algorithm on the first test image

Angle	Worst Run	Best Run
α	45.00°	15.00°
α'	44.93°	14.99°
β	30.00°	0.00°
β'	29.98°	−0.02°
γ	30.00°	60.00°
γ'	28.22°	60.01°
error	1.78°	0.02°

It is worth mentioning that in this example, the auxiliary image $J(\vec{x})$ – the inverse Fourier transform of $|F(\vec{\omega})|^2$ – is indeed different from 0 only in a small vicinity of points which are differences between the component centers; see Figure 2.2.

Need for a second test image. As we have mentioned, for the first test image $I_1(\vec{x})$, the original moments algorithm works just fine, because this image is, in effect, surrounded by an empty space.

The main reason why new algorithms had to be designed is that in practice, images are not always surrounded by an empty space – e.g., in geosciences, a satellite image is just a cut-off of the actual larger image. So, to make a more proper comparison of different algorithms, let us transform the above image $I_1(\vec{x})$ in such a way that it is no longer surrounded by an empty space.

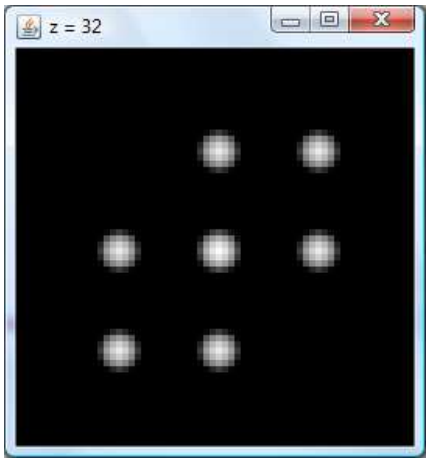


Figure 2.2: The auxiliary function $J(\vec{x})$: inverse Fourier transform of $|F(\vec{\omega})|^2$ for the first test image

How to select a second test image: back-of-the-envelope calculations. To avoid empty space, let us add "noise" to the original image. It is reasonable to expect that when the signal-to-noise ratio is large, the original moments method will still apply, while when the signal-to-noise ratio is small, the original moments method will stop working. A threshold is expected to be when the signal-to-noise ratio is close to 1.

The signal strength can be gauged as the overall intensity S of the first test image $I_1(\vec{x})$, which is $S \approx 6.2$. How can we gauge the noise? Let us first consider a simplified situation when to each intensity value of the original image, we add a normally distributed random noise with 0 mean and standard deviation σ, and that the random values added to the intensity values $I(\vec{x})$ at different points $\vec{x}$ are independent random variables. In this case, the total intensity of this noise is a sum of several independent normally distributed random variables. It is known that this sum is also normally distributed, with 0 mean and the variance σ^2_{total} which is equal to the sum of the variances σ^2 corresponding to all N points $\vec{x}$. Thus, we have $\sigma^2_{\text{total}} = N \cdot \sigma^2$ and $\sigma_{\text{total}} = \sqrt{N} \cdot \sigma$.

In our case, $N = 64^3 = (2^6)^3 = 2^{18}$, so $\sqrt{N} = \sqrt{2^{18}} = 2^9 = 512$. Thus, the signal-to-noise ratio is equal to 1 if $512 \cdot \sigma = S \approx 6.2$, i.e., when $\sigma \approx 0.012$.

To make sure that the original moments method stops working, we need to make sure that the noise actually overwhelms the signal. In engineering applications of statistics, usually, we consider a signal "overwhelming" the normally distributed noise with 0 mean and standard deviation σ if the observed signal is outside the "two sigma interval" $[-2\sigma, 2\sigma]$, i.e., when the signal-to-noise ratio is 2 or larger. Indeed, in this case, the probability that the observed value is actually a noise is below 10%.

In line with this reasoning, we expect that that, vice versa, the noise "overwhelms" the signal if the noise-to-signal ratio exceed 2. Thus, the threshold value is when this ratio is exactly 2, i.e., when

$$\frac{\sigma_{\text{total}}}{S} = \frac{\sqrt{N} \cdot \sigma}{S} = 2$$

and $\sigma = \dfrac{2S}{\sqrt{N}}$. For $S \approx 6.2$ and $N = 64^3$, we get $\sigma \approx \dfrac{2 \cdot 6.2}{512} \approx 0.024$.

Let us thus add a noise of size ≈ 0.024 to all the pixels from the original image $I_1(\vec{x})$.

Comment. The value $\sigma \approx 0.024$ comes from the back-of-the-envelope calculations and is thus only approximate. Our objective is to have simple test images, for which we would be able to easily check the results. In view of this need for simplicity, in the following text we use a simple-to-calculate approximate value $\sigma \approx 0.025$.

Selection of the second test image. As we have mentioned earlier, the simplest way to modify the original image (so as to avoid empty space) is to add a constant k to the original image $I_1(\vec{x})$, i.e., to consider an image

$$I_3(\vec{x}) = I_1(\vec{x}) + k. \tag{2.4}$$

In line with the above reasoning, we select $k = 0.025$.

This value is much smaller than the intensity $A = 1$ of each Gaussian component of the image $I_1(\vec{x})$. As a result, visually, it is practically impossible to distinguish the second test image $I_3(\vec{x})$ from the first one $I_1(\vec{x})$:

In our tests,

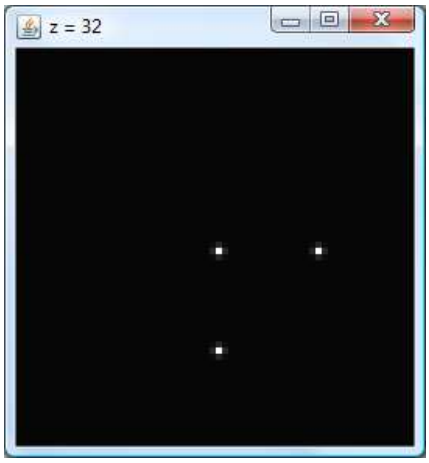


Figure 2.3: The second test image; it is very similar to the first one

- we apply the same shift and angles to this new image $I_3(\vec{x})$ as before, resulting in a new shifted and rotated image $I_4(\vec{x})$, and then
- we apply different referencing algorithms to the images $I_3(\vec{x})$ and $I_4(\vec{x})$ and compare the reconstructed rotation angles with the actual ones.

Comment. In the following text, we also show what happens when we increase the value of k.

Testing the moments of inertia algorithm on the second test image. As we expected, the original moments method does not work for this new image. Not only we find *some* rotation angles which are not reconstructed right, but even the *best-case* reconstruction has an error of about 30 degrees, as the following Table 2.4 shows:

Testing the first new algorithm on the second test image. The first new algorithm was designed for smooth images. It already did not work well for the first test image. The following Table 2.5 shows that its quality is as bad for the second test image:

Table 2.4: The results of testing the moments method on the second test image

Angle	Worst Run	Best Run
α	60.00°	30.00°
α'	−85.63°	53.34°
β	0.00°	15.00°
β'	−7.47°	37.69°
γ	45.00°	0.00°
γ'	7.15°	−26.89°
error	145.63°	26.89°

Table 2.5: The results of testing the first new algorithm on the second test image

Angle	Worst Run	Best Run
α	60.00°	30.00°
α'	72.52°	33.09°
β	60.00°	30.00°
β'	−89.36°	26.64°
γ	0.00°	60.00°
γ'	−53.60°	58.62°
error	149.36°	3.36°

Testing the second new algorithm on the second test image. As expected, the second new algorithm gets much better results. These results are summarized in Table 2.6. In the worst case, we obtain an error of 1.89 degrees, again a reasonable one.

For the second test image $I_3(\vec{x})$, the corresponding function $J(\vec{x}) = FFT^{-1}(|F_3(\vec{\omega})|^2)$ is also similar to what we expect from the above analysis: it is concentrated around the differences of the centers of the image's components; see Figure 2.4.

Table 2.6: The results of testing the second new algorithm on the second test image

Angle	Worst Run	Best Run
α	45.00°	0.00°
α'	44.99°	0.00°
β	30.00°	0.00°
β'	30.16°	0.00°
γ	30.00°	60.00°
γ'	28.11°	60.03°
error	1.78°	0.03°

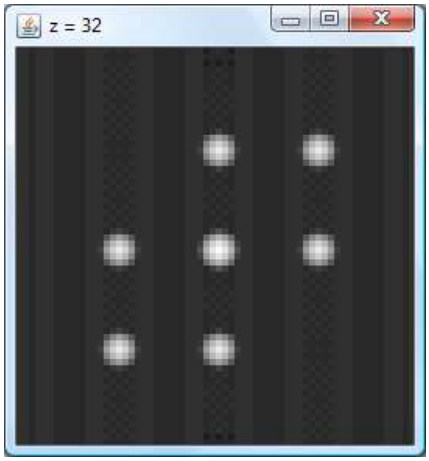


Figure 2.4: The auxiliary function $J(\vec{x})$: inverse Fourier transform of $|F(\vec{\omega})|^2$ for the second test image

Testing the algorithms for larger values of k. The second new algorithm also works for larger values of k; it only starts to fail for $k > 0.1$, when the approximate signal-to-noise ratio $\dfrac{S}{k \cdot \sqrt{N}}$ is ≈ 0.1 (i.e., when the noise is an order of magnitude stronger than the signal). The results of testing the algorithms for different values of k are given in Table 2.7.

Table 2.7: The results of testing the algorithms on the second test image, for different values of the background noise k

k	Original moments method	First new method	Second new method	Image intensity
0.025	145.63°	149.36°	1.89°	6,544
0.050	149.70°	149.36°	2.59°	13,081
0.075	149.70°	149.36°	3.82°	19,662
0.100	149.70°	149.36°	7.75°	26,156
0.125	149.70°	149.36°	125.01°	32,774
0.150	149.70°	149.36°	125.01°	39,319

2.6.2 Testing on Real-Life Images

Pumpkin image: a brief description. As a source of real-life 3-D test images, we took an image of a pumpkin from [185]. Several horizontal "slices" of this image are shown in Figure 2.5.

Pumpkin sub-image used for testing. The original 3-D pumpkin image is surrounded by an empty space, so for this image, the original moments method works well. As we have mentioned, in real life, we often have an image that only covers a part of the object. This partial image is not surrounded by empty space. The main objective of our new algorithms is to process such partial images.

So, to test our new algorithms, we selected such a sub-image of the pumpkin image; namely,

- we first select a $64 \times 64 \times 64$ sub-image starting at position $(120, 120, 14)$;
- then, we cut, from the $64 \times 64 \times 64$ box, a $32 \times 32 \times 32$ box starting at $(16, 16, 16)$.

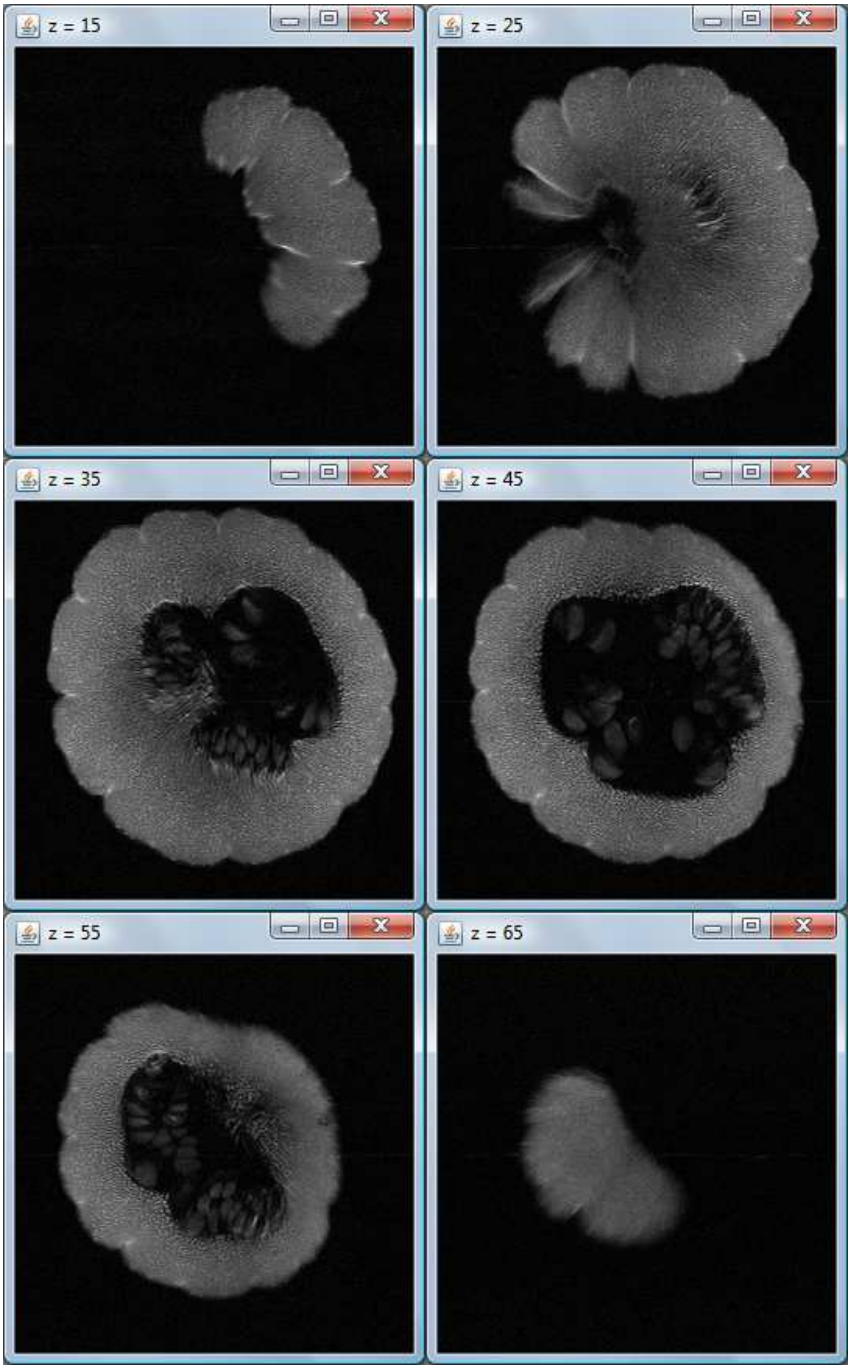


Figure 2.5: The 3-D pumpkin image, shown at several levels z

The pumpkin (sub-)image obtained by this 2-stage procedure will be denoted by $I_{p1}(\vec{x})$.

Shifted and rotated images: description. To test the algorithms, we rotate the $64 \times 64 \times 64$ box by some rotation angles, and then cut a $32 \times 32 \times 32$ box $I_{p2}(\vec{x})$ sub-image from this rotated image, starting at the point $(20, 11, 22)$. The difference in starting points, in effect, produces a shift by $\vec{s} = (4, -5, 6)$.

In our tests, we used 3 possible values of 15, 30, and 45 degrees for each of the rotation angles α, β, and γ. Overall, we tested $3^3 = 27$ combinations of different rotation angles, an amount which is not too small and still allows us to present the results of testing each algorithm on a single page.

Expected testing results. Since the pumpkin image $I_{p1}(\vec{x})$ is not surrounded by empty space, we do not expect the original moments method to succeed in referencing the images $I_{p1}(\vec{x})$ and $I_{p2}(\vec{x})$.

Since the pumpkin sub-image does not consist of abrupt components, we do not expect the second new algorithm to work on this image either.

On the other hand, the pumpkin sub-image is much smoother than the artificial images that we considered so far. We should therefore expect that for this sub-image, the first new algorithm should work.

Our experiments confirm that this is indeed the case: the original moments method and the second new algorithm do not work, while the first new algorithm works. Let us start the description of our results by describing methods which do not work.

Testing the original moments method on the pumpkin sub-image. As we have just argued, since the pumpkin image $I_{p1}(\vec{x})$ is not surrounded by empty space, we do not expect the original moments method to succeed in referencing the images $I_{p1}(\vec{x})$ and $I_{p2}(\vec{x})$.

The results of applying the original moments method to the pumpkin sub-image are shown in Table 2.8; and the image pair that gives the best reconstruction is presented in

Figure 2.6. These results conform that the original moments method does not work for this sub-image: even the best reconstruction of rotation angles is still off by 47.31 degrees.

Table 2.8: Results of testing the original moments method on the pumpkin sub-image

α	β	γ	α'	β'	γ'	error
15.00°	15.00°	15.00°	84.14°	–55.67°	–20.70°	70.67°
30.00°	15.00°	15.00°	–86.14°	–58.94°	–17.39°	116.14°
45.00°	15.00°	15.00°	–79.65°	–65.28°	–20.68°	124.65°
15.00°	30.00°	15.00°	87.24°	–30.48°	–34.05°	72.24°
30.00°	30.00°	15.00°	85.28°	–26.65°	–23.73°	56.65°
45.00°	30.00°	15.00°	79.25°	–17.31°	–19.82°	47.31°
15.00°	45.00°	15.00°	2.77°	51.27°	–80.01°	95.01°
30.00°	45.00°	15.00°	27.80°	53.42°	–57.80°	72.80°
45.00°	45.00°	15.00°	45.95°	50.55°	–40.58°	55.58°
15.00°	15.00°	30.00°	77.82°	–60.51°	–22.74°	75.51°
30.00°	15.00°	30.00°	88.85°	–61.64°	–17.22°	76.64°
45.00°	15.00°	30.00°	–86.86°	–67.60°	–19.04°	131.86°
15.00°	30.00°	30.00°	–38.18°	73.68°	–79.58°	109.58°
30.00°	30.00°	30.00°	88.90°	–35.84°	–23.38°	65.84°
45.00°	30.00°	30.00°	87.95°	–35.06°	–20.88°	65.06°
15.00°	45.00°	30.00°	0.73°	50.71°	–76.95°	106.95°
30.00°	45.00°	30.00°	25.05°	52.52°	–58.19°	88.19°
45.00°	45.00°	30.00°	42.30°	50.28°	–45.20°	75.20°
15.00°	15.00°	45.00°	68.25°	–67.88°	–29.05°	82.88°
30.00°	15.00°	45.00°	84.13°	–63.14°	–18.98°	78.14°
45.00°	15.00°	45.00°	–89.85°	–66.08°	–15.83°	134.85°
15.00°	30.00°	45.00°	29.71°	54.61°	–42.20°	87.20°
30.00°	30.00°	45.00°	79.98°	–7.09°	–17.71°	62.71°
45.00°	30.00°	45.00°	82.13°	–10.54°	–14.71°	59.71°
15.00°	45.00°	45.00°	0.72°	48.09°	–76.51°	121.51°
30.00°	45.00°	45.00°	22.91°	50.38°	–60.73°	105.73°
45.00°	45.00°	45.00°	40.86°	48.02°	–47.69°	92.69°

Testing the second new algorithm on the pumpkin sub-image. As we have mentioned earlier, since the pumpkin sub-image does not consist of abrupt components, we do not expect the second new algorithm to work on this image. Indeed, it does not work.

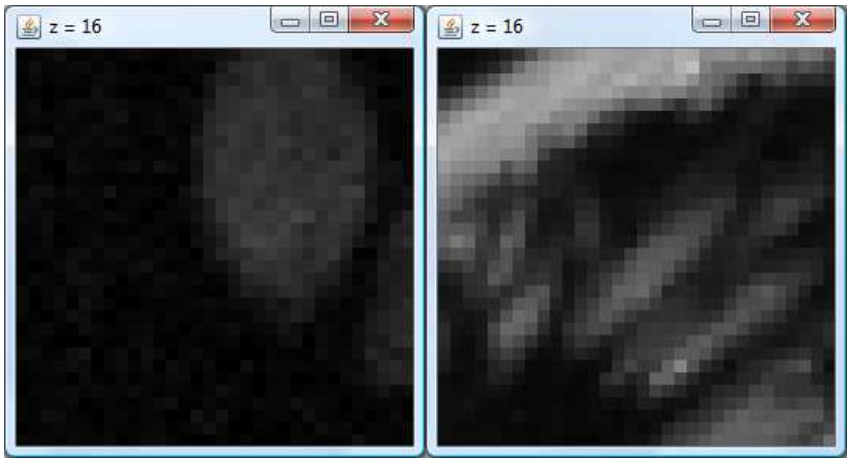


Figure 2.6: Pumpkin sub-images for the rotation angles $\alpha = 45, \beta = 30, \gamma = 15$, at which the original moments method leads to the most accurate reconstruction

The results for applying the second new algorithm to the pumpkin sub-image are shown in Table 2.9; the inverse Fourier transforms $J_{p1}(\vec{x})$ and $J_{p2}(\vec{x})$ of the image pair that leads the best reconstruction are shown in Figure 2.7.

We can see that even the best approximation of rotation angles is still off by 18.73 degrees.

Testing the first new algorithm on the pumpkin sub-image. Finally, let us describe the results of a method which is supposed to work.

As we have mentioned, the pumpkin sub-image is much smoother than the artificial images that we considered so far. We should therefore expect that for this sub-image, the first new algorithm should work.

Results for the first new algorithm are shown in Table 2.10. The algorithm indeed works for this image:

- For 13 of the 27 cases, the error is less than $180/32 = 5.625$ degrees, that is, within one pixel.

Table 2.9: Results of testing the second new algorithm on the pumpkin sub-image

α	β	γ	α'	β'	γ'	error
15.00°	15.00°	15.00°	0.88°	–2.28°	–3.73°	18.73°
30.00°	15.00°	15.00°	13.18°	–1.96°	–4.94°	19.94°
45.00°	15.00°	15.00°	31.06°	–1.60°	–4.88°	19.88°
15.00°	30.00°	15.00°	–2.28°	1.62°	73.10°	58.10°
30.00°	30.00°	15.00°	4.62°	1.81°	–56.52°	71.52°
45.00°	30.00°	15.00°	15.42°	3.98°	–40.45°	55.45°
15.00°	45.00°	15.00°	–3.31°	15.23°	83.51°	68.51°
30.00°	45.00°	15.00°	2.41°	16.44°	–68.30°	83.30°
45.00°	45.00°	15.00°	13.48°	19.68°	–50.97°	65.97°
15.00°	15.00°	30.00°	0.91°	–2.47°	–7.18°	37.18°
30.00°	15.00°	30.00°	11.91°	–2.00°	–6.98°	36.98°
45.00°	15.00°	30.00°	28.52°	–1.86°	–5.59°	35.59°
15.00°	30.00°	30.00°	1.18°	4.19°	–86.76°	116.76°
30.00°	30.00°	30.00°	5.01°	2.46°	–53.24°	83.24°
45.00°	30.00°	30.00°	13.03°	4.07°	–41.81°	71.81°
15.00°	45.00°	30.00°	–2.85°	16.52°	84.43°	54.43°
30.00°	45.00°	30.00°	2.78°	19.40°	–75.08°	105.08°
45.00°	45.00°	30.00°	10.54°	22.39°	–58.84°	88.84°
15.00°	15.00°	45.00°	1.53°	–2.34°	–9.92°	54.92°
30.00°	15.00°	45.00°	12.04°	–1.94°	–9.17°	54.17°
45.00°	15.00°	45.00°	26.96°	–1.59°	–8.00°	53.00°
15.00°	30.00°	45.00°	1.99°	7.29°	–81.12°	126.12°
30.00°	30.00°	45.00°	6.21°	5.72°	–62.15°	107.15°
45.00°	30.00°	45.00°	12.70°	7.07°	–55.47°	100.47°
15.00°	45.00°	45.00°	–2.47°	22.63°	84.02°	39.02°
30.00°	45.00°	45.00°	3.57°	25.71°	–85.29°	130.29°
45.00°	45.00°	45.00°	9.77°	28.46°	–70.24°	115.24°

- For another 7 cases, the error is within 11.25 degrees, or two pixels.
- For three more cases, the error is within 11.875 degrees, or three pixels.

Comment. We have mentioned that the first new algorithm works well when the Fourier transform of an image decreases with frequency. The Fourier transforms of the image pair

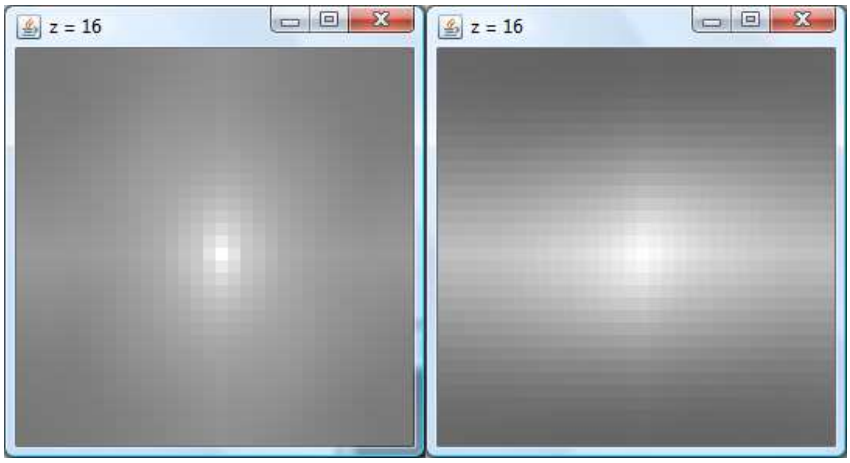


Figure 2.7: Inverse Fourier transforms $|J_{p1}(\vec{x})|$ and $|J_{p2}(\vec{x})|$ of the original and rotated pumpkin sub-images, presented by their midway horizontal planes $z = 16$; rotations by $\alpha = 15$, $\beta = 15$, and $\gamma = 15$ degrees

that leads to the best reconstruction are shown in Figure 2.8. As we can see, the Fourier transform indeed decreases with frequency.

Table 2.10: Results of testing the first new algorithm on the pumpkin sub-image

α	β	γ	α'	β'	γ'	error
15.00°	15.00°	15.00°	18.82°	17.04°	13.59°	3.82°
30.00°	15.00°	15.00°	33.32°	13.44°	19.61°	4.61°
45.00°	15.00°	15.00°	42.61°	14.06°	21.99°	6.99°
15.00°	30.00°	15.00°	18.43°	36.75°	–7.82°	22.82°
30.00°	30.00°	15.00°	32.90°	34.47°	6.36°	8.64°
45.00°	30.00°	15.00°	44.53°	34.89°	18.32°	4.89°
15.00°	45.00°	15.00°	13.44°	41.87°	11.78°	3.22°
30.00°	45.00°	15.00°	31.44°	42.22°	9.45°	5.55°
45.00°	45.00°	15.00°	44.11°	44.01°	17.78°	2.78°
15.00°	15.00°	30.00°	17.12°	14.97°	53.23°	23.23°
30.00°	15.00°	30.00°	33.63°	13.05°	47.64°	17.64°
45.00°	15.00°	30.00°	43.67°	13.25°	31.41°	1.75°
15.00°	30.00°	30.00°	18.94°	38.91°	19.74°	10.26°
30.00°	30.00°	30.00°	34.32°	34.77°	17.97°	12.03°
45.00°	30.00°	30.00°	45.74°	38.70°	23.98°	8.70°
15.00°	45.00°	30.00°	13.81°	44.08°	33.71°	3.71°
30.00°	45.00°	30.00°	30.54°	43.39°	30.71°	1.61°
45.00°	45.00°	30.00°	44.41°	43.81°	32.67°	2.67°
15.00°	15.00°	45.00°	17.64°	17.02°	73.10°	28.10°
30.00°	15.00°	45.00°	32.30°	12.14°	54.92°	9.92°
45.00°	15.00°	45.00°	43.20°	11.52°	42.81°	3.48°
15.00°	30.00°	45.00°	16.92°	38.12°	57.01°	12.01°
30.00°	30.00°	45.00°	33.00°	35.65°	35.35°	9.65°
45.00°	30.00°	45.00°	44.61°	37.51°	29.16°	15.84°
15.00°	45.00°	45.00°	13.46°	45.29°	51.46°	6.46°
30.00°	45.00°	45.00°	31.07°	47.09°	46.43°	2.09°
45.00°	45.00°	45.00°	45.22°	47.64°	40.09°	4.91°

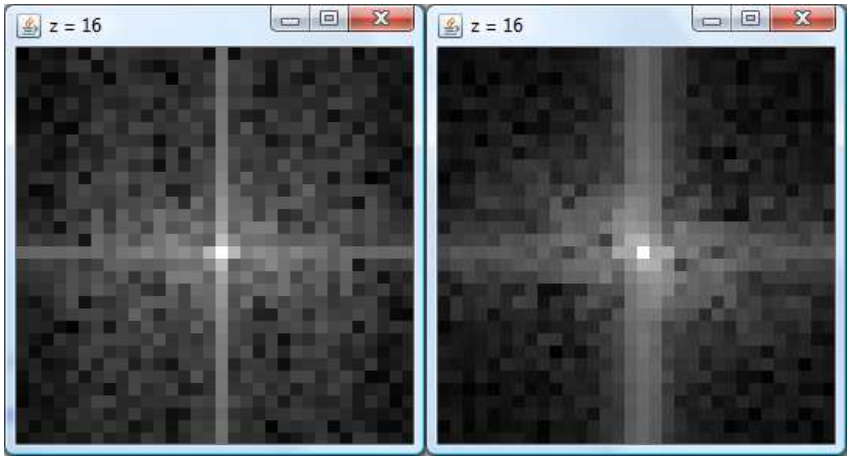


Figure 2.8: Fourier transforms $|F_{p1}(\vec{\omega})|$ and $|F_{p2}(\vec{\omega})|$ of the original and rotated pumpkin sub-images, presented by their midway horizontal planes $z = 16$ (corresponding to $\omega_z = 0$); rotations by $\alpha = 30$, $\beta = 45$, and $\gamma = 30$ degrees

Chapter 3

Using Expert Knowledge in Solving the Seismic Inverse Problem

In the previous chapter, we considered the situation when we can directly measure the image intensity at different points. In this situation, one of the main problems is to reference the corresponding images.

In practice, however, we often encounter more complex data processing situations, in which we cannot measure the image intensity directly. In such situations, we need to reconstruct the image values based on the results of related measurements. An example of such a situation is a seismic inverse problem, where we need to reconstruct the values of the density and the velocity of sound in the Earth from the traveltimes of (natural and artificial) seismic signals. In this chapter, we concentrate on the problem of designing algorithms for such a reconstruction.

Specifically, we will show how to take into account expert knowledge about the domain when designing such algorithms. In particular, we will show how to take into account expert-provided lower bounds $\underline{v}$ and upper bounds $\overline{v}$ on the values of the desired quantity v – i.e., how to take into account interval-related expert knowledge $v \in [\underline{v}, \overline{v}]$.

In this chapter, we will illustrate the possibility to take this interval-related knowledge into account on the example of the seismic inverse problem.

3.1 Seismic Inverse Problem: A Brief Description

In evaluations of natural resources and in the search for natural resources, it is very important to determine Earth's structure. Our civilization greatly depends on the resources we extract from the Earth, such as fossil fuels (oil, coal, natural gas), minerals, and water. Our need for these commodities is constantly growing, and because of this growth, they are being exhausted. Even under the best conservation policies, there is (and there will be) a constant need to find new sources of minerals, fuels, and water.

The only sure-proof way to guarantee that there are resources such as minerals at a certain location is to actually drill a borehole and analyze the materials extracted. However, exploration for natural resources using indirect means began in earnest during the first half of the 20th century. The result was the discovery of many large, relatively easy-to-locate resources such as the oil in the Middle East.

However, nowadays, most easy-to-access mineral resources have already been discovered. For example, new oil fields are mainly discovered either at large depths, or underwater, or in very remote areas – in short, in areas where drilling is very expensive. It is therefore desirable to predict the presence of resources as accurately as possible before we invest in drilling.

From previous exploration experiences, we usually have a good idea of what type of structures are symptomatic for a particular region. For example, oil and gas tend to concentrate near the top of natural underground domal structures. So, to be able to distinguish between more promising and less promising locations, it is desirable to determine the structure of the Earth at these locations. To be more precise, we want to know the structure at different depths z at different locations (x, y).

Determination of Earth's structure also is very important for assessing earthquake risk. Another vitally important application where the knowledge of the Earth's structure is crucial is the assessment of earthquake hazards. Earthquakes can be very destructive, so it is important to be able to estimate the probability of an earthquake, where

one is most likely to occur, and what will be the magnitude of the expected earthquake. Geophysicists have shown that earthquakes result from accumulation of mechanical stress; so if we know the detailed structure of the corresponding Earth locations, we can get a good idea of the corresponding stresses and faults present and the potential for occurrence of an earthquake. From this viewpoint, it also is very important to determine the structure of the Earth.

Data that we can use to determine the Earth's structure. In general, to determine the Earth's structure, we can use different measurement results that can be obtained without actually drilling boreholes: e.g., gravity and magnetic measurements, analyzing the traveltimes and paths of seismic waves as they propagate through the earth.

Forward problem. The relation between the Earth's structure and the related measurable quantities usually is known. So, when we know the exact structure at a given Earth location, we can predict, with reasonable accuracy, the corresponding values of the measured quantities – we can predict the local value of the gravity field, the time that a seismic signal needs to travel from its origin to the sensor, etc.

For example, once we know the density $\rho(\vec{x}) = \rho(x, y, z)$ at different 3-D points (x, y, z), we can determine the resulting ground level ($z = 0$) gravity value $g(\vec{X}) = g(X, Y, 0)$ at a location (X, Y), by using the formula dating back to the original works of Newton:

$$g(\vec{X}) = G \cdot \int \frac{\rho(\vec{x}) \cdot (\vec{x} - \vec{X})}{|\vec{x} - \vec{X}|^3}\, d\vec{x},$$

where G is the universal gravity constant.

In general, corresponding formulas come from problems which geophysicists usually solve: What is the gravity field generated by given mass distribution, what is the magnetic field generated by a given distribution of magnetic materials, what is the traveltime of a seismic signal between two given locations in the given medium, etc. For example, when we want to know how the signal propagates, we can start at the source of the signal and

go forward step-by-step simulating the way the signal actually travels. Such problems are therefore usually called *forward* problems.

Inverse problems. Forward problems enable us, given a model of the Earth, to predict the values of different signals. What we need in the above geophysical applications is the opposite: given the measured values of different signals, we need to reconstruct the structure of the Earth at the location where the measurements have been made. Such problems are therefore called *inverse problems.*

Seismic measurements are usually the most informative. Because of the importance and difficulty of the inverse problem, geophysicists would like to use all possible measurement results: gravity, magnetic, seismic data, etc. In this dissertation, we will concentrate on the measurements which carry the largest amount of information about the Earth's structure and are, therefore, most important for solving inverse problems.

Some measurements – like gravity and magnetic measurements – describe the overall effect of a large area. These measurements can help us determine the average mass density in the area, or the average concentration of magnetic materials in the area, but they often do not determine a detailed structure of this area. This detailed structure can be determined only from measurements which are narrowly focused on small sub-areas of interest.

The most important of these measurements are usually *seismic measurements*. Seismic measurements involve the recording of vibrations caused by distant earthquakes, explosions, or mechanical devices. For example, these records are what seismographic stations all over the world still use to detect earthquakes. However, the signal coming from an earthquake carries not only information about the earthquake itself, it also carries the information about the materials along the path from an earthquake to the station: e.g., by measuring the traveltime of a seismic wave, checking how fast the signal came, we can determine the speed of sound v in these materials.

Terminological comment. In geosciences, the speed of sound is usually called the *velocity* of sound, or simply *velocity*. Since our main case study is the seismic inverse problem in geophysics, in this chapter, we will use the terms "velocity of sound" or "velocity".

Usually, the velocity of sound increases with increasing density, so, by knowing the velocity of sound at different 3-D points, we will be able to determine the density of materials at different locations and different depths.

The main problem with the analysis of earthquake data (i.e., *passive* seismic data) is that earthquakes are rare events, and they mainly occur in a few seismically active belts. Thus, we have a very uneven distribution of sources and receivers that results in a "fuzzy" image of Earth's structure in many areas.

To get a better understanding of the Earth's structure, we must therefore rely on *active* seismic data – in other words, we must make artificial explosions, place sensors around them, and measure how the resulting seismic waves propagate. The most important information about the seismic wave is the *traveltime* t_i, i.e., the time that it takes for the wave to travel from its source to the sensor. To determine the geophysical structure of a region, we measure seismic travel times and reconstruct velocities at different depths from these data. The problem of reconstructing this structure is called the *seismic inverse problem*.

3.2 How to Describe the Seismic Inverse Problem in Precise Terms: Current Approaches and Their Limitations

Need to find velocities $v(\vec{x})$ at different points. Let us start by a brief summary of the geophysical problem. One of the main objectives of geophysics is to determine the 3-D structure of the Earth. Different crustal materials usually have different density and, as a result, different velocity of sound v. Therefore, one way to determine the desired structure

is to determine the velocity $v(\vec{x})$ at different 3-D points $\vec{x} = (x_1, x_2, x_3)$.

Data for determining the velocities. To find the desired velocities $v(\vec{x})$, we set up explosions at different locations on the Earth surface, place sensors around these locations, and measure the time for the resulting seismic wave to propagate to the sensors.

Based on these measured traveltimes t_i and known locations of sensors, we then reconstruct the velocities. The problem of reconstructing the velocities is called the *seismic inverse problem*.

Usual assumption. In the seismic inverse problem, it is usually assumed that each seismic wave follows the shortest path between the source and the sensor. This assumption can be viewed as a model of the seismic wave propagation.

Precise formulation of the seismic inverse problem: first try. In view of the above assumption, the seismic inverse problem can be formulated as follows:

- We are given a number n (number of sensor readings).
- For each i from 1 to n, we know a spatial point S_i (location of the source), a spatial point R_i (location of the sensor), and the travel time t_i.
- We need to find a function $v(\vec{x})$ such that for each i,

$$\min_{\gamma: S_i \to R_i} \int_\gamma \frac{ds}{v(\vec{x})} = t_i,$$

  where the minimum is taken over all possible paths γ which start at the point S_i and end at the point R_i.

Comment. In reality, the above assumption is only approximate because it assumes that the rays travel along a single path and ignores the effects of *diffraction*, when a wave "spreads out" beyond the path. However, in most instances of the seismic inverse problem, we can safely ignore this spread.

Un-resolved issue with the above formulation: incompleteness. A serious issue with the above formalization of the inverse problem is its incompleteness. Specifically, the traveltimes only carry information about the locations through which the seismic waves pass. Many 3-D points $\vec{x}$ are not covered by the corresponding ray paths. As a result, based only on the traveltimes and on the shortest-path model, we cannot make any conclusions about the value $v(\vec{x})$ of the velocity at these points.

How the incompleteness problem is usually resolved: smoothness assumptions and regularization. Incompleteness is typical in numerical mathematics problems. Usually, it is resolved by making additional assumptions such as an assumption of smoothness (differentiability) – that the desired functions (such as our $v(x)$) smoothly depends on the coordinates. This smoothness assumption is behind the *regularization* techniques for solving the corresponding problems.

In the seismic inverse problem, there is no smoothness. In the seismic inverse problem, it is known that the dependence $v(x)$ is, in reality, not smooth: there exist layers of different rocks, with an abrupt (discontinuous) transition between them. As a result, in general, we may have different layered solutions each of which is consistent with the same measurement results.

Some possible solutions are geophysically meaningful, and some are not. From the viewpoint of the above formulation of the seismic inverse problem – via an incomplete model – all these drastically different solutions should be considered satisfactory. However, in practice, when a geophysicist expert looks at these solutions, he or she is normally able to select one (or a few) of these solutions which are consistent with the expert's understanding of geophysics – and dismiss other solutions as geophysically meaningless.

It is desirable to describe "geophysically meaningful" in precise terms. Since experts can distinguish between geophysically meaningful and geophysically meaningless

solutions, it is desirable to formally describe their ability to do it – so that the resulting formalized expert information could be used to "complete" the incomplete model.

How to describe "geophysically meaningful": an ideal situation. In the ideal world, we should be able to formulate the expert knowledge in a formalized way and thus, replace the original incomplete model with a (more) complete one – that combines the original equations with the formalized expert knowledge.

How experts actually select "geophysically meaningful": an "algorithm-selection" approach. In many problems like the seismic inverse problem, we are still very far away from formalization of the expert knowledge. Instead, the following "algorithm-selection" approach is used to describe geophysically meaningful solutions.

Specifically, the fact that we can have many drastically different solutions to each instance of this problem means that we can have many different algorithms which produce a solution for each such instance. In general, for the same instance, these algorithms lead to different solutions. The essence of the "algorithm-selection" approach is that among all the algorithms which have been proposed and tested, experts select an algorithm which seems to be providing, in general, the best solutions. The selected algorithm is then used to solve all the instances of the problem.

The results of the selected algorithm are better not because they are in better accordance with the original (incomplete) model – all compared algorithms are in approximately the same good level of accordance. The selected algorithm is "better" because it better reflects the expert knowledge.

In this sense, the algorithm (method) becomes, in effect, a part of our model of the physical reality – the part which reflects expert knowledge. So, to describe the additional expert knowledge, we must provide the algorithm selected by the corresponding expert community.

Comment. Selection of the best algorithm is an ongoing process. Once the best-so-far algorithm is selected based on its prior performance, it often happens that on some later instances, the results produced by the selected algorithm are not in good accordance with the experts' understanding. In such situation, the selected algorithm has to be modified; sometimes, a completely new algorithm has to be designed which then becomes the selected one.

Selected algorithm for solving the seismic inverse problem. In the seismic inverse problem, Hole's code [89] is the algorithm that has been selected by the geophysics community and which is now most widely used to solve this problem.

Comment. This algorithm will be described, in detail, in the next section.

Limitations of the algorithm-selection approach to formalizing the seismic inverse problem. As we have mentioned, in general, the selected algorithm is usually not perfect, it is only the best of the few which have been proposed. Crudely speaking, we select a method that is, *on average*, the best reflection of the expert knowledge, and we hope that this method provides a geophysically meaningful solution $v(\vec{x})$ in *all* (or at least in *most*) practical instances of the seismic inverse problem. Sometimes, the selected method does lead to a meaningful solution, but in many practical situations, the resulting solution is not in full agreement with the expert's intuition.

Need to consider interval-related expert knowledge: a reminder. As we have mentioned earlier, in some situations, a part of the expert's knowledge can be explicitly formulated. For example, a geophysicist may know that the velocity at a certain depth must lie within certain reasonable bounds, e.g., between 5 and 8 km/s. At present, in such situations, researchers try to repeatedly modify ("hack") the process until the results produced by the algorithm agree with this expert knowledge. This process takes up a lot of expert time and – because of the need for numerous iterations – a lot of computer

time. To avoid this long, ad-hoc process, it is desirable to explicitly incorporate the expert knowledge into the algorithms, so that the results are always consistent with the expert's knowledge.

Expert knowledge usually comes in the form of bounds (i.e., intervals) on the actual values of the physical quantities. It is therefore desirable to take this interval-related expert knowledge into account when processing data.

3.3 Known Algorithms for Solving the Seismic Inverse Problem: Description, Successes, Limitations

The main objectives of this section. The main objective of this chapter is to incorporate interval-related expert knowledge into the existing algorithms for solving the seismic inverse problem. In other words, our objective is to *modify* the existing algorithm so that its results will be in better agreement with this knowledge (e.g., the resulting values $v(\vec{x})$ should lie within the corresponding expert-provided intervals $[\underline{v}, \overline{v}]$).

To be able to meaningfully modify the existing algorithm, we must understand where this algorithm comes from, exactly what objective function is optimizes – so that, hopefully, the optimization of this objective function under interval constraints $v(\vec{x}) \in [\underline{v}, \overline{v}]$ will lead to the desired modification.

Let us therefore describe, in detail, the origins of Hole's algorithm, the algorithm selected by the geoscientists for solving the seismic inverse problem.

First step towards understanding and describing Hole's algorithm: discretization. Our objective is to find a function $v(\vec{x})$. In general, we need infinitely many parameters to describe all such functions: e.g., the values $v(\vec{x})$ corresponding to all (infinitely many) points $\vec{x}$. In practice, we can only generate finitely many parameters, so we must restrict ourselves to some finite-parametric family of functions – i.e., we must *discretize* the problem.

In numerical mathematics, one of the most widely used discretization technique is the finite elements approach. In this approach, we divide the 3-D domain into sub-domains (finite elements), and use simple expressions to describe the behavior of the velocity $v(\vec{x})$ on each of these elements. In geophysics, usually, a rectangular grid is used to generate such finite elements.

How can we describe the behavior of function on a given element? For a smooth function $f(\vec{x})$, we may approximate it by a constant, or we may get a more accurate approximation if we use linear or quadratic (or higher-order polynomial) functions to approximate the behavior of $f(\vec{x})$ on each element.

We have already mentioned that for the seismic inverse problem, the desired function $v(\vec{x})$ is, in general, not smooth. For such functions, piecewise linear or piecewise quadratic (smoothing) approximation may lead to a geophysically misleading picture, in which instead of an abrupt transition between several geophysical layers we have a smooth one. To avoid such misleading solutions, geophysicists normally approximate the behavior of a function $v(\vec{x})$ on each element by a constant – i.e., use a piecewise constant approximation.

In other words, we divide the 3-D zone of interest into a grid of rectangular elements (cells), and we assume that the velocity of sound is constant throughout each cell.

To describe all possible solutions of this type, we must describe, for each cell j, the velocity v_j in this cell.

How to reformulate the seismic inverse problem in terms of the velocities v_j. Once we know the velocities v_j in each cell j, we can then determine the paths which seismic waves take. Seismic waves travel along the shortest path – shortest in terms of time. It can be easily determined that for such paths, within each cell, the path is a straight line, and on the border between the two cells with velocities v and v', the direction of the path changes in accordance with Snell's law

$$\frac{\sin(\varphi)}{v} = \frac{\sin(\varphi')}{v'},$$

where φ and φ' are the angles between the paths and the line orthogonal to the border between the cells. (If this formula leads to $\sin(\varphi') > 1$, this means that this wave cannot penetrate into the neighboring cell at all; instead, it bounces back into the original cell with the same angle φ.)

In particular, we can thus determine the paths from the source to each sensor. The traveltime t_i along the i-th path can then be determined as the sum of traveltimes in different cells j through which this path passes: $t_i = \sum_j \frac{\ell_{ij}}{v_j}$, where ℓ_{ij} denotes the length of the part of the i-th path within cell j.

This formula becomes easier to describe if we replace the original unknowns – velocities v_j – by their inverses $s_j \stackrel{\text{def}}{=} \frac{1}{v_j}$, called *slownesses*. In terms of slownesses, the formula for the traveltime takes the simpler form $t_i = \sum_j \ell_{ij} \cdot s_j$.

We arrive at a non-linear problem. The objective of the seismic inverse problem is to reconstruct the unknown values of the slowness s_j from the known traveltimes t_i.

At first glance, the above equations may look like a system of linear equations for the unknowns s_j, but in reality, the system is strongly non-linear, because the lengths ℓ_{ij} of the shortest paths depend on the corresponding slownesses $s_1, s_2, \ldots$. In other words, a more accurate reformulation of this equation should take the following (explicitly nonlinear) form $t_i = \sum_j \ell_{ij}(s_1, s_2, \ldots) \cdot s_j$.

Hole's algorithm for solving the seismic inverse problem: motivations and derivation. At each stage of the Hole's algorithm, we have some approximation to the desired slownesses. We start with some reasonable initial slownesses, and we hope that after several iterations, we will be able to get slownesses which are much closer to the actual values.

At each iteration, we first use the currently known slownesses s_j to find the corresponding paths from the source to each sensor. Based on these paths, we compute the predicted values $t_i = \sum_j \ell_{ij} \cdot s_j$ of traveltimes.

Since the currently known slownesses s_j are only approximately correct, the traveltimes t_i (which are predicted based on these slownesses) are approximately equal to the measured traveltimes $\widetilde{t}_i$; there is, in general, a discrepancy $\Delta t_i \stackrel{\text{def}}{=} \widetilde{t}_i - t_i \neq 0$. It is therefore necessary to use these discrepancies to update the current values of slownesses, i.e., replace the current values s_j with corrected values $s_j + \Delta s_j$. The objective of this correction is eliminate (or at least decrease) the discrepancies $\Delta t_i \neq 0$. In other words, the objective is to make sure that for the corrected values of the slowness, the predicted traveltimes are closer to $\widetilde{t}_i$.

Of course, once we have changed the slownesses, the shortest paths will also change; however, if the current values of slownesses are reasonable, the differences in slowness are not large, and thus, paths will not change much. Thus, in the first approximation, we can assume that the paths are the same, i.e., that for each i and j, the length ℓ_{ij} remains the same. In this approximation, the new traveltimes are equal to $\sum \ell_{ij} \cdot (s_j + \Delta s_j)$. The desired condition is then $\sum \ell_{ij} \cdot (s_j + \Delta s_j) = \widetilde{t}_i$. Subtracting the formula $t_i = \sum\limits_j \ell_{ij} \cdot s_j$ from this expression, we conclude that the corrections Δs_j must satisfy the following system of (approximate) linear equations: $\sum \ell_{ij} \cdot \Delta s_j \approx \Delta t_i$.

Solving this system of linear equations is not an easy task, because we have many observations and many cell values and thus, many unknowns. For a generic system of linear equations, standard methods for solving it (e.g., Gauss elimination methods) require computation time which grows as a cube n^3 of the number of variables n. Our system is not generic, it is sparse – for every traveltime i, we have $\ell_{ij} \neq 0$ only for the cells j on the corresponding path. For sparse matrices, there exist faster algorithms for solving the corresponding systems of linear equations; see, e.g., [219]. However, these algorithms still require a large amount of computation time. So, instead of the standard methods for solving a system of linear equations, most geoscientists use special faster geophysics-motivated techniques (described below) for solving the corresponding systems. These methods are described, in detail, in the next subsection.

Once we solve the corresponding system of linear equations, we compute the updated values Δs_j, compute the new (corrected) slownesses $s_j + \Delta s_j$, and repeat the procedure

again. We stop when the discrepancies become small; usually, we stop when the mean square error $\frac{1}{n}\sum_{i=1}^{n}(\Delta t_i)^2$ no longer exceeds a given threshold. This threshold is normally set up to be equal to the measurement noise level, so that we stop iterations when the discrepancy between the model and the observations falls below the noise level – i.e., when, for all practical purposes, the model is adequate.

Algorithm for the inverse problem: details. Let us describe, in more detail, how the corresponding linear system of equations usually is solved. In other words, for a given cell j, how do we find the correction Δs_j to the current value of slowness s_j in this cell?

Let us first consider the simplified case when there only is one path, and this path is going through the j-th cell. In this case, cells through which this path does not go does not need any correction. To find the corrections Δs_j for all the cells j through which this path goes, we only have one equation $\sum\limits_j \ell_{ij} \cdot \Delta s_j = \Delta t_i$. The resulting system of linear equations is clearly under-determined: we have a single equation to find the values of several variables Δs_j. Since the system is under-determined, we have a infinite number of possible solutions. Our objective is to select the most geophysical reasonable of these solutions.

For that, we can use the following idea. Our single observation involves several cells; we cannot distinguish between the effects of slownesses in different cells, we only observe the overall effect. Therefore, there is no reason to assume that the value Δs_j in one of these cells is different from the values in other cells. It is thus reasonable to assume that all these values are equal to each other: $\Delta s_j = \Delta s_k = \ldots = \Delta s$. Substituting these equal values into the equation $\sum\limits_j \ell_{ij} \cdot \Delta s_j = \Delta t_i$, we conclude that $L_i \cdot \Delta s = \Delta t_i$, where $L_i = \sum\limits_j \ell_{ij}$ is the overall length of i-th path. Thus, in the simplified case in which there is only one path, to the slowness of each cell j along this path, we add the same value $\Delta s = \dfrac{\Delta t_i}{L_i}$.

Let us now consider the realistic case in which there are many paths, and moreover, for many cells j, there are many paths i which go through the corresponding cell. For a given cell j, based on each path i passing through this cell, we can estimate the correction Δs_j by the corresponding value $\Delta s_{ij} \stackrel{\text{def}}{=} \dfrac{\Delta t_i}{L_i}$. When there are several (n_j) paths P_i going through

the j-th cell C_j ($P_i \cap C_j \neq \emptyset$), we have, in general, several different estimates $\Delta s_j \approx \Delta s_{ij}$.

Now, we have an over-determined system of equations: we have a single variable Δs_j and as many equations (n_j) as there are paths i going through this cell. Ideally, we would like all the differences $e_i \stackrel{\text{def}}{=} \Delta s_j - \Delta s_{ij}$ to be equal to 0, i.e., we would like the corresponding difference vector $e = (e_i, e_{i'}, \ldots)$ to be equal to 0. In general, the estimates Δs_{ij} are different, so we cannot find a single correction Δs_j which is equal to all of them – and thus, we cannot have all the components of the difference vector e to be equal to 0. Since we cannot make e *exactly* equal to 0, it is reasonable to find the value Δs_j for which the corresponding vector e is as *close* to 0 as possible, i.e., for which the distance $\sqrt{s_i^2 + s_{i'}^2 + \ldots}$ between e and $0 = (0, 0, \ldots)$ is the smallest possible.

Since the square root is a monotonic function, the square root of the expression is the smallest if and only if the expression itself is the smallest. Thus, minimizing the distance is equivalent to minimizing the sum $s_i^2 + s_{i'}^2 + \ldots$ So, we arrive at the Least Squares method for solving a system of over-determined equations. For the system $\Delta s_j \approx \Delta s_{ij}$ the solution is well known: the arithmetic average of different estimates:

$$\Delta s_j = \frac{1}{n_j} \sum_{P_i \cap C_j \neq \emptyset} \frac{\Delta t_i}{L_i}.$$

This is the formula used in Hole's algorithm.

Comment about convergence. A natural question is whether this algorithm converges. Since the problem is highly nonlinear, and the algorithm is rather complicated, there are no known convergence results. Empirically, it has been shown that this algorithm sometimes diverges, but this divergence occurs when we select a wrong first approximation. Usually, if we start with a first approximation which is geophysically reasonable, the algorithm converges in a few iterations (converges in the sense that after few iterations, the corresponding velocity values v_j barely change).

Modifications of Hole's algorithm. The above formula treats all the paths passing through a given cell equally. In reality, some paths may barely touch the cell, while other

paths really traverse the entire cell. The paths which barely touch the cell should not have the same influence on the cell as the others. How can we achieve this objective?

In the expression $\Delta t_i = \sum_j \ell_{ij} \cdot \Delta s_j$, the value Δs_j occurs with a coefficient ℓ_{ij}. Thus, for the same accuracy in traveltimes Δt_i, the resulting accuracy with which we can determine Δs_j from i-th path is proportional to $1/\ell_{ij}$. In other words, the accuracy σ_i with which $\Delta s_j \approx \Delta s_{ij}$ is proportional to $1/\ell_{ij}$. By applying the Least Square Method to this new system of equations, we conclude that $\sum_i \frac{e_i^2}{\sigma_i^2} \to \min$, i.e., that $\sum_i (\Delta s_j - \Delta s_{ij})^2 \cdot \ell_{ij}^2 \to \min$. Differentiating over Δs_j, we conclude that

$$\Delta s_j = \frac{\sum_j \Delta s_{ij} \cdot \ell_{ij}^2}{\sum_j \ell_{ij}^2},$$

i.e., that instead of a *simple* average, we take a *weighted* average of estimates coming from different paths i, with weights proportional to ℓ_{ij}^2. This method has also been used in solving the inverse problem; it uses slightly more computations on each iteration but, since iterations are somewhat more reasonable, requires fewer iterations.

Similar more sophisticated algorithms are described, e.g., in [169, 238].

Successes of the known algorithms. The known algorithms have been actively used to reconstruct the slownesses, and, in many practical situations, they have led to reasonable geophysical solutions $v(\vec{x})$.

Limitations of the known algorithms. As we have mentioned, the inverse problem is often under-determined: We have fewer observations than unknowns. As a result, based on the same measurement results, we may have many different velocity models $v_j = 1/s_j$ that are all consistent with the same measurement results.

The above algorithm selects one of these velocity models; often, however, the velocity model that is returned by the existing algorithm is not geophysically meaningful: E.g., it predicts velocities outside of the range of reasonable velocities at this depth. To get a

geophysically meaningful model, a geophysicist tries to adjust the initial approximation – so as to avoid this discrepancy between the actual distribution and the geophysical knowledge.

This adjustment usually requires several iterations. It is a very time-consuming process, because there is no algorithmic way of adjusting the initial data, only heuristic recipes, and as a result, each adjustment requires many time-consuming trial-and-error steps. Moreover, because of the non-algorithmic character of adjustment, it requires special difficult-to-learn skills; as a result, the existing tools for solving the seismic inverse problem are not as widely used as they could be.

3.4 How to Use Interval-Related Expert Knowledge

Main idea. As we have mentioned, one of the reasons why the mathematically valid solution is not geophysically meaningful is that at some points, the velocity of sound is outside the interval of values which are possible at this depth for this particular geological region.

Additional interval-related information provided by experts. To take this expert knowledge into consideration, it is reasonable to explicitly solicit, from the experts, the information about possible values of slownesses – and then modify the inverse algorithms in such a way that the velocities are consistent with this knowledge.

Specifically, for each cell j, a geophysicist provides us with his or her bounds $\underline{s}_j$ and $\overline{s}_j$ on the slowness s_j. In other words, for every cell j, we know that the slowness s_j belongs to the corresponding interval $[\underline{s}_j, \overline{s}_j]$.

Solution to the seismic inverse problem under interval uncertainty. For each cell j, we have an interval $[\underline{s}_j, \overline{s}_j]$ that is guaranteed to contain the actual (unknown) value of slowness s_j.

We select the initial model $s_j^{(0)}$ that is consistent with this information, i.e., for which $s_j^{(0)} \in [\underline{s}_j, \overline{s}_j]$ for all j. Then, we must modify the algorithm for solving the inverse prob-

lem in such a way that on all iterations, slownesses always stay within the corresponding intervals.

How to use interval information: first idea. To explain our first idea, let us reformulate the problem. Suppose that we start with the slowness values $s_j^{(0)}$ which are within the given intervals: $s_j^{(0)} \in [\underline{s}_j, \overline{s}_j]$. On the next iteration, however, we may get values $s_j^{(0)} + \Delta s_j$ which are outside the corresponding intervals.

For example, we may know the exact values of slownesses in the upper layers (closer to the surface), and these are exactly the values which we select as the initial approximation for the corresponding cells. Since we know the exact slownesses for the surface cells, for paths which only go through these surface cells, the predicted traveltimes are close to the measured ones – so no correction is needed. However, for paths which do go through deeper cells as well, we will, in general, need a correction. The existing algorithms evenly spread the resulting correction Δs to all the cells along the path. So, if we follow these algorithms, then, in addition to the deeper cells, we also update slownesses in the upper layer cells. This update changes the slowness values and thus, leads to values outside the given (degenerate) interval $[s_j, s_j]$.

To remedy this situation, we can do the following. For each cell j, after an iteration of, say, Hole's algorithm, we have a corrected value of the slowness $s_j^{(k)}$ which approximates the actual (unknown) slowness s_j: $s_j \approx s_j^{(k)}$. We also know that s_j should be located in the interval $[\underline{s}_j, \overline{s}_j]$. Similar to our previous analysis, it is therefore reasonable to use the Least Squares Method to combine these two piece of information: i.e., we look for the value $s_j \in [\underline{s}_j, \overline{s}_j]$ for which the square $(s_j - s_j^{(k)})^2$ is the smallest possible. In geometric terms, we look for the value within the given interval $[\underline{s}_j, \overline{s}_j]$ which is the closest to $s_j^{(k)}$. Thus:

- If the value $s_j^{(k)}$ is already within the interval, we keep it intact.
- If the value $s_j^{(k)}$ is to the left of the interval, i.e., if $s_j^{(k)} < \underline{s}_j$, then the closest point from the interval is its left endpoint $\underline{s}_j$.

- Similarly, if the value $s_j^{(k)}$ is to the right of the interval, i.e., if $s_j^{(k)} > \overline{s}_j$, then the closest point from the interval is its right endpoint $\overline{s}_j$.

Algorithm resulting from the first idea. We thus arrive at the following modification of the existing iterative algorithms for solving the seismic inverse problem. This modification takes into account that for each cell j, we know an interval $[\underline{s}_j, \overline{s}_j]$ which is guarantee to contain the actual (unknown) value of slowness s_j.

We start with an initial approximation $s_j^{(0)}$ which is consistent with this information, i.e., for which $s_j^{(0)} \in [\underline{s}_j, \overline{s}_j]$ for all j. On each iteration, we first apply the iteration step from the existing algorithms, and then update the resulting values $s_j^{(k)}$ as follows:

- if $s_j^{(k)} < \underline{s}_j$, we replace the value $s_j^{(k)}$ with $\underline{s}_j$;
- if $s_j^{(k)} > \overline{s}_j$, we replace the value $s_j^{(k)}$ with $\overline{s}_j$;
- if $\underline{s}_j \le s_j^{(k)} \le \overline{s}_j$, we keep the value $s_j^{(k)}$.

After this additional step, we perform the next iteration, etc.

Advantage of the first idea. The main advantage of this approach is that on every iteration, we get slownesses which are within the given intervals. Thus, in contrast to the existing algorithms, we always remain within the given intervals and thus, avoid non-physical solutions.

First idea: main problem. The main problem with this approach is that it is still too slow. Let us explain why it may be even slower than the traditional algorithms. We will illustrate this on the same example on which we explained, above, why the un-modified Hole's algorithm can lead us outside the desired interval for s_j. In this example, Hole's algorithm equally distributes the discrepancy Δt_i between all the cells along the i-th path: both surface cells and the deeper cells, as $\Delta s = \Delta t_i / L_i$. After the slowness adjustment of all these cells, we change the original predicted traveltime value of $t_i = \sum_j \ell_{ij} \cdot s_j$ to the

new value $t_i + \sum\limits_j \ell_{ij} \cdot \Delta s_j = t_i + \Delta s \cdot L_i = t_i + \Delta t_i$, i.e., to $\widetilde{t}_i$. Thus, on the next iteration, the discrepancy along this path will be much smaller (or even disappear completely).

In the new algorithm, after producing the same correction $\Delta s = \Delta_i / L_i$, we, in effect, dismiss it for all the upper-layer cells – because for these cells, adding this correction will bring us outside the given (degenerate) interval. As a result, after the correction, the resulting addition to the traveltime comes only from the deeper cells, and is thus equal to $\ell_i \cdot \Delta s$, where ℓ_i is the overall length of all deep portions of the i-th path. Since the path also covers several upper-layer cells, we have $\ell_i < L_i$, thus $\ell_i \cdot \Delta s < L_i \cdot \Delta s = \Delta t_i$; so, a large portion of discrepancy re-appears in the next iteration.

Second idea. To speed up computations, it is desirable not just to dismiss the slowness corrections that lead us outside the given intervals $[\underline{s}_j, \overline{s}_j]$, but also to re-distribute the traveltime discrepancy that remains uncovered as a result of this dismissal. In other words, instead of simply *repeating* each iteration step from Hole's algorithm, we *modify* these steps so as to make computations faster and still remain within given slowness intervals.

On each iteration of the new procedure, we start with the slowness values $s_j^{(k-1)}$ which are within given intervals $[\underline{s}_j, \overline{s}_j]$, and we want to produce corrected values $s_j^{(k-1)} + \Delta s_j$ within these same intervals. The condition $s_j^{(k-1)} + \Delta s_j \in [\underline{s}_j, \overline{s}_j]$ is equivalent to $\Delta s_j \in [\underline{\Delta}_j, \overline{\Delta}_j]$, where $\underline{\Delta}_j \stackrel{\text{def}}{=} \underline{s}_j - s_j^{(k-1)}$, and $\overline{\Delta}_j \stackrel{\text{def}}{=} \overline{s}_j - s_j^{(k-1)}$.

If the values Δs_{ij} corresponding to each path i are within the desired interval $[\underline{\Delta}_j, \overline{\Delta}_j]$, then their average is also inside this same interval – and, if we want to use the weighted average instead (as mentioned above), this weighted average is also guaranteed to be within the given interval. Thus, to guarantee that the resulting average corrections are within the given intervals, it is sufficient, for each path i, to produce the distribution of slowness corrections Δs_{ij} which are within the corresponding intervals $[\underline{\Delta}_j, \overline{\Delta}_j]$.

Let us therefore concentrate on a single path i. To better describe the idea, let us assume that $\Delta t_i > 0$ (the case $\Delta t_i < 0$ is similar). In this case, our idea is as follows:

- First, we distribute the time discrepancy Δt_i by the overall length L_i of i-th path,

and get the correction value $\Delta s_{ij} = \Delta t_i / L_i$.

- Then, we check whether for all the cells j along the path i, we have $\Delta s_{ij} \le \overline{\Delta}_j$:
  - if this inequality is satisfied for all cells j, then we stop iterations and return the results;
  - otherwise, if there are some cells j for which the correction $\Delta s_{ij} = \Delta t_i / L_i$ exceeds the allowed maximum correction $\overline{\Delta}_j$, then for these cells, we use the maximum correction instead of the values $\Delta t_i / L_i$.
- We compute the traveltime change $t_i^{(1)} \stackrel{\text{def}}{=} \sum\limits_j \ell_{ij} \cdot \overline{\Delta}_j$ resulting from these slowness corrections; here, the sum is taken only over the cells for which the slowness correction is fixed at the value $\overline{\Delta}_j$; the set of such cells will be denoted by $I^{(1)}$.
- We then compute the remaining discrepancy $\Delta t_i^{(1)} \stackrel{\text{def}}{=} \Delta t_i - t_i^{(1)}$ – which needs to be distributed among the remaining cells along the i-th path.
- Then, we divide the remaining discrepancy $\Delta t_i^{(1)}$ among the remaining cells, producing the value $\Delta s_{ij}^{(1)} = \Delta t_i^{(1)} / L_i^{(1)}$, where $L_i^{(1)} = \sum\limits_{j \notin I^{(1)}} \ell_{ij}$ is the total path within the remaining cells.
- We check whether for all the remaining cells j, we have $\Delta s_{ij}^{(1)} \le \overline{\Delta}_j$:
  - if this inequality is satisfied for all cells j, then we stop iterations and return the results;
  - otherwise, if there are some cells for which the correction $\Delta t_i^{(1)} / L_i^{(1)}$ exceeds the allowed maximum correction $\overline{\Delta}_j$, for these cells, we use the maximum correction instead of the values $\Delta t_i / L_i$; let us denote the set of such cells by $I^{(2)}$.
- We compute the traveltime change $t_i^{(2)} \stackrel{\text{def}}{=} \sum\limits_{j \in I^{(2)}} \ell_{ij} \cdot \overline{\Delta}_j$ resulting from these slowness corrections.

- We then compute the remaining discrepancy $\Delta t_i^{(2)} \stackrel{\text{def}}{=} \Delta t_i^{(1)} - t_i^{(2)}$ – which needs to be distributed among the remaining cells along the i-th path, etc.

Comment. In general, the above algorithm leads to the assignment of slowness corrections which stay within the desired intervals – and, at the same time, still cover the discrepancy Δt_i. Sometimes, however, even after picking each slowness correction at its highest allowed level $\overline{\Delta}_j$, we will still not cover the observed time discrepancy Δt_i: i.e., $\sum\limits_j \ell_{ij} \cdot \overline{\Delta}_j < \Delta t_i$. This means that the observed traveltimes are *inconsistent* with the intervals $[\underline{s}_j, \overline{s}_j]$. This information should be reported back to the experts, so that the experts will be able to adjust their bounds for s_j in such a way that the new bounds will be consistent with the observations.

Advantage of the second idea. The main problem with the *first* idea was that it while it guaranteed the slownesses within the given intervals, this idea required, in general, more iterations than the original algorithm. The reason for this increase is as follows:

- in the original algorithm, the discrepancy at the next iteration is much smaller than in the previous one;
- in contrast, in the first idea, the discrepancies may decrease only slightly.

When we use the *second* idea, then on each iteration, we decrease all the discrepancies Δt_i as much as possible. As a result, on the next iteration, the discrepancy (if any) is much smaller than the original one. Thus, the number of iterations is about the same as for the original algorithm – and at the same time, we always get slownesses within the given interval.

So, in terms of the number of iterations, the algorithm based on the second idea is faster than the algorithm based on the first idea – and comparable with the number of iterations in the traditional algorithm for solving seismic inverse problems.

Drawbacks of the second idea. The main drawback of the second idea is that while the *number* of iterations goes down, the number of necessary computations *within* each iteration drastically increases.

Indeed, for each path i, in the original Hole's algorithm, all we need to do is compute the overall length L_i along the i-th path, divide Δ_i by L_i, and then add the resulting slowness correction Δs to all the slowness values. Computing the length L_i – by adding the lengths ℓ_{ij} of the segments within different cells j – requires as many arithmetic operations as there are such cells; let us denote this number by c. Division is 1 operation, and adding Δs to all c slownesses also requires c arithmetic operations. Thus, overall, we need $c + 1 + c = O(c)$ operations.

For the algorithm based on the second idea, we still need $O(c)$ steps in the first round, but we may then need second, third, etc., rounds. On each round, at least one of the new slowness correction is assigned to one of the cells; thus, the number of rounds cannot exceed the overall number of cells c along the path. On the other hand, it is possible that we will need as many rounds as there are cells. So, in the worst case, we need c rounds with $O(c)$ operations in each – to the total of $O(c^2)$ arithmetic operations.

In other words, each iteration may require $O(c^2)$ steps instead of $O(c)$. Paths can be long, up to 100-200 km, and cells can be of 1 km size, so we can have c in hundreds. For such paths, a c times increase (= hundreds times increase) can drastically increase the computation time of each iteration and thus, drastically increase the overall computation time. It is thus desirable to come up with a faster method of reallocating the traveltime discrepancy. Let us describe such a method.

How to use interval information: third idea. We want to find the corrections $\Delta s_j \in [\underline{\Delta}_j, \overline{\Delta}_j]$ for which $\sum\limits_j \ell_{ij} \cdot \Delta s_j = \Delta t_i$. Similarly to the above derivation of the original Hole's algorithm, the resulting system is clearly under-determined: we have a single equation to find the values of several variables Δs_j. Since the system is under-determined, we have an infinite number of possible solutions, so we must select the most geophysical reasonable of

these solutions.

Similarly to the above derivation of Hole's algorithm, the idea is that there is no reason to assume that the value Δs_j in one of these cells is different from the values in other cells, so we assume that $\Delta s_j \approx \Delta s_{j'}$ for all j and j'. In Hole's algorithm, we simply assumed that all these approximate equalities are actual equalities, so all the values Δs_j along the path were equal, but that assumption leads to slownesses outside given interval ranges. So, in general, some of these equalities can only be approximately true. As we mentioned in our derivation of Hole's code, this situation can be naturally handled by using the Least Squares Method, i.e., in this case, by selecting, from all the solutions of the above system of equations and inequalities, the values Δs_j for which the objective function $\sum\limits_{j \neq j'} (\Delta s_j - \Delta s_{j'})^2$ takes the smallest possible value.

For simplicity, we can add values $j = j'$ in the sum without changing the value of the objective function – since, for $j = j'$, the corresponding differences $\Delta s_j - \Delta s_{j'}$ are 0s anyway. This objective function can be further simplified if we introduce the notations $E \stackrel{\text{def}}{=} \dfrac{1}{c} \sum\limits_j \Delta s_j$ and $\delta_j \stackrel{\text{def}}{=} \Delta s_j - E$, so that $\sum\limits_j \delta_j = 0$. In these notations, $\Delta s_j - \Delta s_{j'} = \delta_j - \delta_{j'}$, so

$$\sum_{j,j'} (\delta_j - \delta_{j'})^2 = \sum_j \delta_j^2 + \sum_{j'} \delta_{j'}^2 - 2 \cdot \sum_j \sum_{j'} \delta_j \cdot \delta_{j'}.$$

The last term is the product of the two 0 sums $\left(\sum\limits_j \delta_j \right) \cdot \left(\sum\limits_{j'} \delta_{j'} \right)$, and the first two terms contain $2c^2$ expressions δ_j^2 – these two terms cover each of c squares δ_j^2 the same number of times (namely, $2c^2/c = 2c$ times). Thus, the objective function is equal to $2c \cdot \sum\limits_j \delta_j^2$. So, minimizing the objective function is the same as minimizing the population variance $V = \dfrac{1}{c} \cdot \sum\limits_j \delta_j^2$ of the population Δs_j.

In the case when we do not have the linear constraint $\sum\limits_j \ell_{ij} \cdot \Delta s_j = \Delta t_i$, there exist efficient algorithms for minimizing variance under interval uncertainty in time $O(c \cdot \log(c))$ (which is faster than $O(c^2)$); see, e.g., [81, 118]. Let us show that these algorithms can be modified to the case when have the linear constraint $\sum\limits_j \ell_{ij} \cdot \Delta s_j = \Delta t_i$.

Third idea: towards a faster algorithm. Let us consider the case $\Delta t_i > 0$. The case $\Delta t_i < 0$ can be treated similarly.

If we have $\Delta s_j \in [\underline{\Delta}_j, \overline{\Delta}_j)$ for two different values j and j' for which $\Delta s_j \neq \Delta s_{j'}$, then we can replace both values Δs_j and $\Delta s_{j'}$ by their weighted average $\dfrac{\ell_{ij} \cdot \Delta s_j + \ell_{ij'} \cdot \Delta s_{j'}}{\Delta s_j + \Delta s_{j'}}$ without changing Δt_i, and the variance will only decrease. Similarly, if for some j, we have $\Delta s_j < \overline{\Delta}_j$, and for some other $j' \neq j$, we have $\Delta s_{j'} = \overline{\Delta}_{j'} > \Delta s_j$, then we can replace both values by their weighted average and thus, decrease the variance.

Thus, when the variance attains its minimum, all the values Δs_j which are not "maxed out" to $\overline{\Delta}_j$ are equal to each other, and all the maxed-out values do not exceed the common non-maxed-out value. Let p denote the overall number of the values Δs_j which are maxed out when the variance is minimized. So, if we sort the bounds $\overline{\Delta}_1, \ldots, \overline{\Delta}_c$ into a non-decreasing sequence

$$\overline{\Delta}_{(1)} \leq \overline{\Delta}_{(2)} \leq \ldots \leq \overline{\Delta}_{(c)},$$

then the smallest value of the variance is attained when $\Delta s_{(1)} = \overline{\Delta}_{(1)}$, $\Delta s_{(2)} = \overline{\Delta}_{(2)}$, $\ldots$, $\Delta s_{(p)} = \overline{\Delta}_{(p)}$, and $\Delta s_{(p+1)} = \ldots = \Delta s_{(c)} = \delta$. The common non-maxed-out value δ can be found from the condition that $\sum\limits_j \ell_{ij} \cdot \Delta s_j = \Delta t_i$, i.e., that $A_p + \mathcal{L}_p \cdot \delta = \Delta t_i$, where

$$A_p \stackrel{\text{def}}{=} \sum_{j=1}^{p} \ell_{i(j)} \cdot \overline{\Delta}_{(j)} \text{ and } \mathcal{L}_p \stackrel{\text{def}}{=} \sum_{j=p+1}^{c} \ell_{i(j)}.$$

Therefore, $\delta = \dfrac{\Delta t_i - A_p}{\mathcal{L}_p}$. Thus, once we know the value p, we can easily compute the corresponding slowness corrections Δs_j. The only remaining problem is how to find the value p.

To find p, we must use the fact that since only p values Δs_j are maxed out, the $(p+1)$-st value $\Delta s_{(p+1)}$ is not maxed out, so $\delta = \Delta s_{(p+1)} < \overline{\Delta}_{(p+1)}$. Hence, from the above equation for Δt_i, we conclude that $\Delta t_i < S_p \stackrel{\text{def}}{=} A_p + \mathcal{L}_p \cdot \overline{\Delta}_{(p+1)}$. (For $p = c$, a similar inequality holds if we take $\overline{\Delta}_{c+1)} = \infty$.)

Similarly, from the fact that the non-maxed-out value δ should be larger than all the maxed-out ones, we conclude that $\delta \geq \overline{\Delta}_{(p)}$, hence $\Delta t_i \geq A_p + \mathcal{L}_p \cdot \overline{\Delta}_{(p)}$. Let us simplify

this condition. By definition,

$$A_p + \mathcal{L}_p \cdot \overline{\Delta}_{(p)} = \sum_{j=1}^{p} \ell_{i(j)} \cdot \overline{\Delta}_{(j)} + \sum_{j=p+1}^{c} \ell_{i(j)} \cdot \overline{\Delta}_{(p)}.$$

By moving the term proportional to $\overline{\Delta}_{(p)}$ from the first into the second sum, we conclude that

$$A_p + \mathcal{L}_p \cdot \overline{\Delta}_{(p)} = \sum_{j=1}^{p-1} \ell_{i(j)} \cdot \overline{\Delta}_{(j)} + \sum_{j=p}^{c} \ell_{i(j)} \cdot \overline{\Delta}_{(p)} = A_{p-1} + \mathcal{L}_{p-1} \cdot \overline{\Delta}_{(p)} = S_{p-1}.$$

So, the two conditions on Δt_i take the form $S_{p-1} \le \Delta t_i < S_p$.

It is easy to check that for every p, when we go from S_{p-1} to S_p, then for every j, the coefficient at $\ell_{i(j)}$ can only increase, so $S_{p-1} \le S_p$. Thus, the sequence $S_0, \ldots, S_c$ is non-decreasing: $S_0 \le S_1 \le S_2 \le \ldots \le S_c$, and p can be found as the only value for which Δ_i belongs to the corresponding subinterval $[S_{p-1}, S_p)$. So, we arrive at the following algorithm for computing slowness corrections for each path.

The resulting new algorithm for interval uncertainty. We start with the initial slowness values $s_j^{(0)}$ which are within the given intervals $[\underline{s}_j, \overline{s}_j]$.

On each iteration of the new procedure, we start with the slowness values $s_j^{(k-1)}$ which are within given intervals $[\underline{s}_j, \overline{s}_j]$. We then compute, for each cell j, the values $\underline{\Delta}_j = \underline{s}_j - s_j^{(k-1)}$ and $\overline{\Delta}_j = \overline{s}_j - s_j^{(k-1)}$.

Based on these slownesses, we find the paths from the sources to the sensors, compute the predicted traveltimes t_i along each path, and the discrepancies $\Delta t_i = \widetilde{t}_i - t_i$.

Let us describe how we compute the correction Δs_j along the i-th path. Once we have computed these corrections for all the paths, then for each cell j, we take the average (or weighted average) of all the corrections coming from all the paths which pass through this cell.

We will consider the case when $\Delta t_i > 0$; the case when $\Delta t_i < 0$ is treated similarly. In this case, we first sort all c values $\overline{\Delta}_j$ along the i-th path into a non-decreasing sequence

$$\overline{\Delta}_{(1)} \le \overline{\Delta}_{(2)} \le \ldots \le \overline{\Delta}_{(c)}.$$

Then, for every p from 0 to c, we compute the values A_p and $\mathcal{L}_p$ as follows:

$$A_0 = 0, \quad \mathcal{L}_0 = L_i, \quad A_p = A_{p-1} + \ell_{i(p)} \cdot \overline{\Delta}_{(p)}, \quad \mathcal{L}_p = \mathcal{L}_{p-1} - \ell_{i(p)}.$$

After that, for each p, we compute $S_p = A_p + \mathcal{L}_p \cdot \Delta_{(p+1)}$, and we find p for which $S_{p-1} \le \Delta t_i < S_p$. Once this p is found, we take $\Delta s_{(j)} = \overline{\Delta}_j$ for $j \le p$, and for $j > p$, we take $\Delta s_{(j)} = \dfrac{\Delta t_i - A_p}{\mathcal{L}_p}$.

When $\Delta t_i < 0$, we similarly sort the values $\underline{\Delta}_j$ into a decreasing sequence, and find p so that the first p corrections are "maxed out" to $\underline{\Delta}_j$, and the rest $c - p$ corrections are determined from the condition $\Delta s_{(j)} = \dfrac{\Delta t_i - A_p}{\mathcal{L}_p}$.

Computational complexity of the new algorithm. Let us show that this new algorithm is indeed faster.

- Sorting c values requires time $O(c \cdot \log(c))$.
- Computing each of c values A_p, $\mathcal{L}_p$, and S_p requires a finite number ($O(1)$) of elementary arithmetic operations, so the overall computation time for this step is $O(c)$.
- Finally, once p is found, we need a finite number of step to compute each of c slowness corrections Δs_j, so we also need $O(c)$ steps.

Overall, we thus need $O(c \cdot \log(c)) + O(c) + O(c) = O(c \cdot \log(c))$ steps.

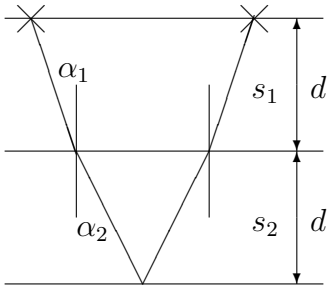
3.5 Example Showing that the Use of Expert Knowledge Drastically Improves the Performance of Algorithms for Solving the Seismic Inverse Problem

Let us show, on a simple example in which we will be able to analytically trace all the computations, that the new algorithm is indeed faster than the original Hole's algorithm. In this example, we have two vertical layers of the same height d. The lighter rocks

eventually lift up, so, closer to the surface, the densities are usually smaller, hence, the sound velocities are lower and the slownesses are larger. Thus, we assume that $s_1 > s_2$.

To avoid the need to consider deeper layers, we assume that the structure below the second layer is so heavy that all the signals simply bounce back from the bottom of the second layer (in real geological situations, this is what happens, e.g., at the Moho surface).

To be able to analytically handle ray tracing, we restrict ourselves to the situations when in the Snell's law, all the angles α_i are small ($\alpha_i \ll 1$), and thus, we can safely assume that $\sin(\alpha_i) \approx \alpha_i$.



The deeper the layers, the more difficult it is to explore them. As a result, we usually have more information about the upper layers than about the deeper layers. In accordance with this general idea, in our simplified example, we assume that we know the exact value of slowness s_1 at the upper layer, but we only know an approximate value $\widetilde{s}_2 \approx s_2$ of the slowness at the lower layer. For simplicity, we can assume that the value $\widetilde{s}_2$ is close to the actual value s_2 – so that in our computations, we can safely ignore terms which are quadratic in $\Delta s_2 \stackrel{\text{def}}{=} \widetilde{s}_2 - s_2$.

For simplicity, we will only consider one signal. (Actually, in this simplified example, the conclusion will be the same if we consider several signals.) Both for the original Hole's code and for the new layer, we consider the values s_1 and $\widetilde{s}_2$ as the first approximation. In the new algorithm, we also impose the interval constraint $[s_1, s_1]$ on the slowness in the upper layer. We will show that in this example, the new algorithm is better in the following sense: in the new algorithm, after a single iteration, the difference between the

current approximations (s_1, s_2') and the actual (unknown) slownesses decreases practically to 0, while in the original Hole's iterative method, this difference, while decreasing, stays within the same order of magnitude. For example, if $\alpha_i \approx 0.1$, we get 10 times error decrease in the new methods and only ≈ 2 times decrease in the original one.

Let us perform the corresponding computations. Let α_1 and α_2 be the angles of the ray corresponding to the actual (unknown) slownesses s_1 and s_2, and let $\widetilde{\alpha}_1 = \alpha_1 + \Delta\alpha_1$ and $\widetilde{\alpha}_2 = \alpha_2 + \Delta\alpha_2$ be the angles obtained by ray tracing from the approximate values s_1 and $\widetilde{s}_2$.

The horizontal distance between the source and the sensor can be computed as the sum of the four horizontal distances corresponding to the four parts of the path, i.e., as $2d \cdot (\tan(\alpha_1) + \tan(\alpha_2))$. Since the angles are small, we have $\tan(\alpha_i) \approx \alpha_i$, so this distance takes the form $2d \cdot (\alpha_1 + \alpha_2)$. We should get the same distance whether we consider the original slownesses (and the corresponding angles α_i), or computed slownesses and the corresponding angles $\widetilde{\alpha}_i = \alpha_i + \Delta\alpha_i$. Thus, we conclude that $\alpha_1 + \alpha_2 = (\alpha_1 + \Delta\alpha_1) + (\alpha_2 + \Delta\alpha_2)$, i.e., that $\Delta\alpha_2 = -\Delta\alpha_1$.

To find the relation between $\Delta\alpha_i$ and the slownesses, we must use Snell's law. Since we assume that the angles are small and $\sin(\alpha_i) \approx \alpha_i$, Snell's law for the actual angles takes the form $s_1 \cdot \alpha_1 = s_2 \cdot \alpha_2$, and Snell's law for the computed angles takes the form $s_1 \cdot \widetilde{\alpha}_1 = \widetilde{s}_2 \cdot \widetilde{\alpha}_2$, i.e., $s_1 \cdot (\alpha_1 + \Delta\alpha_1) = (s_2 + \Delta s_2) \cdot (\alpha_2 + \Delta\alpha_2)$. Subtracting these two equations from each other, and ignoring second-order terms like $\Delta s_2 \cdot \Delta\alpha_2$, we conclude that $s_1 \cdot \Delta\alpha_1 = s_2 \cdot \Delta\alpha_2 + \Delta s_2 \cdot \alpha_2$. Since $\Delta\alpha_2 = -\Delta\alpha_1$, we thus get $\Delta\alpha_1 \cdot (s_1 - s_2) = \Delta s_2 \cdot \alpha_2$, hence

$$\Delta\alpha_1 = \Delta s_2 \cdot \frac{\alpha_2}{s_1 - s_2}.$$

The actual path in the first layer is equal to $\ell_1 = \dfrac{2d}{\cos(\alpha_1)}$, the actual path in the second layer is equal to $\ell_2 = \dfrac{2d}{\cos(\alpha_2)}$. Since we assume that the angles α_i are small and we can ignore quadratic and higher order terms in terms of α_i, we conclude that $\cos(\alpha_i) \approx 1$ and $\ell_1 = \ell_2 = 2d$. Thus, the actual traveltime is equal to $s_1 \cdot \ell_1 + s_2 \cdot \ell_2 = 2d \cdot (s_1 + s_2)$.

For simplicity, we can assume that traveltimes are measured with high accuracy, hence the measured traveltime $\widetilde{t}$ is equal to $2d \cdot (s_1 + s_2)$.

For the computed trajectory, we similarly conclude that the computed travel time is equal to $t = 2d \cdot (s_1 + \widetilde{s}_2) = 2d \cdot (s_1 + s_2 + \Delta s_2)$. Thus, the discrepancy in the traveltimes is equal to $\Delta t = \widetilde{t} - t = -2d \cdot \Delta s_2$.

In the original Hole's algorithm, we divide this discrepancy by the entire path $L = \ell_1 + \ell_2 = 4d$, and add the resulting value $-\dfrac{\Delta s_2}{2}$ to both slownesses. As a result, we replace the original approximate slowness $\widetilde{s}_2 = s_2 + \Delta s_2$ with a more accurate estimate $\widetilde{s}_2 = s_2 + \dfrac{\Delta s_2}{2}$. In other words, the approximation error decreases by a factor of two. This is not bad at all: if we can decrease the approximation error by half in each iteration, then in, say, 7 iterations we reduce the original approximation error to less than 1% of the original one. However, as we will see, the new method leads to an even faster convergence.

Indeed, in the new method, since the value s_1 is already at the endpoint of the corresponding interval $[s_1, s_1]$, we divide the traveltime discrepancy $\Delta t = -2d \cdot \Delta s_2$ only by the length $\ell_2 = 2d$ of the second part of the path. As a result, according to the new algorithm, to the original estimate $\widetilde{s}_2 = s_2 + \Delta s_2$, we add the correction $\dfrac{\Delta t}{\ell_2} = \dfrac{-2d \cdot \Delta s_2}{2d} = -\Delta s_2$. After adding this correction, we get the new value $(s_2 + \Delta s_2) - \Delta s_2 = s_2$.

In other words, while in the original Hole's algorithm, after a single iteration, the error decreased in half, in the new algorithm, the approximation error – in this first approximation – disappears completely.

In reality, the approximation error does not disappear completely: In our computations, we ignored terms of higher order in terms of α_i. In other words, all the above equalities are actually equalities modulo terms proportional to α_i. Thus, if, e.g. $\alpha_i \approx 0.1$, then we do not get exactly 0 error, we may get a 0.1 of the original error. Still, it means that in a single iteration, we get a 10 times decrease in approximation error and thus, our convergence is much faster than for the original Hole's code where the error decreases only 2 times in each iteration.

3.6 Computational Complexity of the Seismic Inverse Problem and Why Traditional Methods of Solving Inverse Problems Do Not Work Well in the Seismic Case

Motivation. In the above sections, we use sophisticated techniques to come up with approximate solutions to the original problem. Moreover, we used methods like Hole's algorithm for which even convergence is not always guaranteed. To justify the use of such heuristic approximate techniques, let us show that the corresponding problems are indeed computationally difficult – namely, NP-hard.

Most inverse problems in science and engineering are ill-posed. The above ill-posedness of the seismic problem is a common feature in applications: most inverse problems in science and engineering are ill-posed; see, e.g., [213].

Smoothness: traditional approach to solving ill-posed inverse problems. A typical way to solve an inverse problem is to find a natural physically meaningful property of actual solution, and use this *a priori* information to select a single most physically meaningful solution among many mathematically possible ones. This process is called *regularization.*

Typically, in inverse problems, this natural property is smoothness. Smoothness can be naturally described in precise mathematical terms. For example, when we reconstruct a 1-D signal $x(t)$, then the degree of smoothness can be defined as follows. At a given moment of time t, the larger the absolute value $|x'(t)|$ of the derivative $x'(t)$, the less smooth the signal is. Thus, at a given time t, the value $|x'(t)|$ is a natural degree of the signal's non-smoothness. Overall, a natural degree of non-smoothness can be defined as a mean square of these degrees corresponding to different moments t, i.e., as $J \stackrel{\text{def}}{=} \int (x'(t))^2\,dt$.

Most regularization techniques try to find, among many signals that are consistent with given observations, the smoothest signal, i.e., the signal with the smallest possible value of the degree of non-smoothness J.

Smoothness: discrete case. In real life, we only have the values

$$x(t_1), x(t_2), \ldots,$$

of the signal $x(t)$ at discrete moment of time

$$t_1, t_2 = t_1 + \Delta t, \ldots, t_{i+1} = t_i + \Delta t, \ldots$$

Based on this discrete data, we can approximate the derivative $x'(t)$ as a difference

$$\frac{x(t_{i+1}) - x(t_i)}{\Delta t},$$

so minimizing the integral J is equivalent to minimizing the corresponding integral sum

$$J_{\text{discr}} \stackrel{\text{def}}{=} \sum_i (x(t_{i+1}) - x(t_i))^2.$$

Smoothness: 2-D case. For a 2-D velocity distribution $f(n_1, n_2)$, similarly, a natural assumption is that this distribution is smooth. Similarly to the 1-D case, a natural way to describe the degree of smoothness of a given distribution is to use the integral sum

$$J \stackrel{\text{def}}{=} \sum_{n_1, n_2} s(n_1, n_2),$$

where

$$s(n_1, n_2) \stackrel{\text{def}}{=}$$

$$(f(n_1 + 1, n_2) - f(n_1, n_2))^2 + (f(n_1, n_2 + 1) - f(n_1, n_2))^2.$$

Alternatively, we can describe this criterion as the sum of the squares of the differences in intensity between all possible pairs (p, p') of neighboring pixels $p = (n_1, n_2)$ and $p' = (n_1', n_2')$:

$$J = \sum_{p, p' \text{ are neighbors}} (f(p) - f(p'))^2.$$

Smoothness makes problems computationally solvable. A practically useful property of the above degrees of non-smoothness J is that the expression J is a convex function of the signal $x(t_i)$ or $f(n_1, n_2)$. Thus, if the conditions describing the fact that the unknown velocity distributions is consistent with the observations is also described by linear or, more generally, smooth inequalities, then the problem of finding the regularized solution can be reformulated as a problem of minimizing a convex function J on the convex set.

Similarly, if we fix the degree of non-smoothness and look, among all the solutions with a given degree of non-smoothness, for the one that is the closest to the original approximate solution, we also have a problem of minimizing a convex function (distance) on the convex set (of all functions that are consistent with the observations and have the desired degree of smoothness).

It is known that, in general, the problems of minimizing convex functions over convex domains are algorithmically solvable (see, e.g., [220]), and smoothness-based regularization has indeed been efficiently implemented; see, e.g., [213].

For the seismic inverse problem, we only have piecewise smoothness. In geophysics, we have clear layers of different types of rocks, with sharp difference between different layers, so we face an inverse problem with only piecewise smoothness; see, e.g., [169].

Traditional smoothness measures are not adequate for piecewise smoothness. In the piecewise smooth case, the above measure of non-smoothness is not applicable, because it would include neighboring pixels on the different sides of the border between the two layers.

Appropriate smoothness measures for piecewise smoothness case. To avoid the above problem, we need to only take into account the pairs of neighboring pixels that

belong to the same zone (layer), i.e., consider the sum

$$J(Z) = \sum_{p \sim p' \text{ are neighbors in the same zone}} (f(p) - f(p'))^2,$$

where Z denotes the information about the zones, and $p \sim p'$ means that p and p' are neighbors in the same zone. This measure makes computational sense only if we know beforehand where the zones are – i.e., where is the border between the two zones.

However, in real life, finding the border is a part of the problem. In this case, we can use the same smoothness criterion not only to reconstruct the original velocity distribution, but also to find the border location. Specifically, we want to look for the zone distribution *and* for the zone location for which the above criterion J takes the smallest possible value.

In other words, we fix the number of zones, and we characterize the non-smoothness of an velocity distribution by a criterion

$$J^* = \min_Z J(Z),$$

where the minimum is taken over all possible divisions Z into zones.

The resulting problem is no longer convex. The resulting functional is no longer convex, because the division into zones is a discrete problem. It is known that non-convex problems are, in general, more computationally difficult than the corresponding convex ones (see, e.g., [98]), and adding discrete variables makes the problems even more computationally difficult; see, e.g., [168].

Complexity of piecewise smooth inverse problems. In the following sections, we show that in general, the inverse problem for piecewise smooth case is computationally intractable (NP-hard) even when the relation expressing the consistency between the measured results and the desired velocity distribution is linear.

This proof will follow the proof of NP-hardness of different signal processing problems described, e.g., in [34, 106].

Let us prove that in general, the inverse problem for piecewise smooth case is computationally intractable (NP-hard).

Main idea of the proof: reduction to a subset problem. To prove NP-hardness of our problem, we will reduce a known NP-hard problem to the problem whose NP-hardness we try to prove: namely, to the inverse problem for piecewise smooth velocity distributions.

Specifically, we will reduce, to our problem, the following *subset sum* problem [106, 168] that is known to be NP-hard:

- Given:
  - m positive integers $s_1, \ldots, s_m$ and
  - an integer $s > 0$,
- check whether it is possible to find a subset of this set of integers whose sum is equal to exactly s.

For each i, we can take $x_i = 0$ if we do not include the i-th integer in the subset, and $x_i = 1$ if we do. Then the subset problem takes the following form: check whether there exist values $x_i \in \{0, 1\}$ for which

$$\sum s_i \cdot x_i = s.$$

We will reduce each instance of this problem to the corresponding piecewise smooth inverse problem.

Reduction to a subset problem: details. Let us consider the following problem. We want to reconstruct an $m \times m$ velocity distribution $f(n_1, n_2)$. Let $d = \lfloor m/2 \rfloor$. We want a piecewise smooth velocity distribution $f(n_1, n_2)$ that consists of two zones.

The following linear constraints describe the consistency between the observations and the desired velocity distribution:

- $f(n_1, n_2) = 1$ for $n_2 > d$;

- $\sum_{i=1}^{m} s_i \cdot f(i,d) = s$; and
- $f(n_1, n_2) = 0$ for $n_2 < d$.

The problem that we consider is to find the solution with the smallest possible value of smoothness J^* among all the velocity distributions that satisfy these linear constraints.

Let us show that the minimum of J^* is 0 if and only if the original instance of the subset problem has a solution.

Indeed, if J^* is 0, this means that all the values within each zone must be the same. Since we have values 1 for $n_2 > d$ and values 0 for $n_2 < d$, we must therefore have every value to be equal either to 0 or to 1. Thus, if we have such a solution, the corresponding values $f(i,d) \in \{0,1\}$ provide the solution to the original subset problem $\sum s_i \cdot x_i = s$.

Vice versa, if the selected instance of the original subset problem has a solution x_i, then we can take $f(i,d) = x_i$ and get the solution of the inverse problem for which the degree of non-smoothness is exactly 0.

So, if we can solve the inverse problem for piecewise smooth velocity distributions, we will thus be able to solve the subset sum problem.

This reduction proves that the inverse problem for piecewise smooth velocity distributions is indeed NP-hard.

Chapter 4

Interval Methods in Estimating How Close Is the Solution to the Inverse Problem to the Actual Image: Case Study

4.1 Need to Estimate How Close Is the Solution to the Inverse Problem to the Actual Image

In the previous chapter, we explained that interval methods are needed to incorporate expert knowledge into the solution of an inverse problem. When solving an inverse problem, we also need to estimate how close the resulting solution is to the actual image. In this dissertation, we show how interval methods can be used to estimate this closeness, and we illustrate these methods on a simple example.

4.2 Main Reasons Why the Solution to the Inverse Problem Is Different from the Actual Image: In Brief

In order to gauge how close the solution is to the actual image, we need to recall the main reasons why the solution is different from the actual image.

Differences caused by modeling errors. First, as we have mentioned in Chapter 1, the solution is based on a model for the desired real-life objects and processes. Models are always approximate; there is always a *modeling error*. Since the solution is based on a model which describes the real-life processes only approximately, this solution itself can only be an approximation to the actual image. As we have mentioned, for good models, this error is usually small, but it still needs to be taken into account.

This modeling error is usually estimated during the process of validating the model, a process that precedes the solution to the inverse problem.

One of the main reasons why there often is a substantial modeling error is that we do not have enough information about the real-life process. In principle, the more resources we have, the more data we can gather, and we will thus get a model which is more and more adequate in describing the real objects and processes.

Differences caused by the incompleteness of data. Second, there is an additional source of the difference between the solution and the actual image: the *incompleteness* of the measurement results. For example, in the seismic inverse problem, if we knew the exact travel times between every two points, we would be able to determine the velocity of sound $v(x)$ at different 3-D points x. In practice, we only know the travel times between finitely many pairs of points; as a result, some 3-D points y are not covered by the corresponding ray paths. Therefore, based only on the available travel times, we cannot uniquely determine the velocity $v(y)$ at these points. So, the value $v(y)$ from the solution is, in general, different

from the actual value of the velocity of sound at this point. This difference is very difficult to gauge, because gauging it requires a deep expertise in the corresponding area of knowledge.

Differences caused by the approximate character of the numerical algorithms. Third, the existing methods of solving the corresponding equations are approximate. One of the main reasons why the solutions are approximate is that the actual images can be reasonably arbitrary functions (e.g., smooth or piece-wise smooth). In general, to describe such a function exactly, we need to know the exact values of infinitely many parameters: e.g., the exact values of the actual image $v(x)$ at infinitely many points x, or the exact values of its Fourier transform at infinitely many different spatial frequencies. Inside the computer, however, we can only describe and process finitely many real numbers, and for each of these real numbers, we can only store finitely many bits describing this real number. In other words, in order to solve the problem, we *discretize* the actual image (i.e., approximate it by images from a finite-parametric family), and we *approximate* the actual values of the corresponding parameters by close computer-representable numbers. These discretization and approximation are one more reason why the resulting solution is, in general, different from the actual image.

This difference depends on our selection of the number of parameters and of the accuracy with which we represent and process each of these parameters. In other words, this difference depends on the actual program implementation of the corresponding numerical method. One of the main objectives of numerical mathematics is to find, within given computational restrictions, a numerical methods which provides the best possible accuracy – i.e., for which this component of the difference between the solution and the actual image is as small as possible. As a result, in numerical mathematics, there are many methods to estimate this *accuracy* – and to improve this accuracy. In other words, this component of the difference is among the most well-studied.

Difference caused by the uncertainty of the input data. In addition to the above three sources of the difference between the numerical solution to the inverse problem and the actual image, there is yet another reason for the difference between the solution and the actual value.

Even when the model is perfect, the measurements are practically complete, and the numerical algorithms provide an almost exact solution to the corresponding equations, we still have an uncertainty. Indeed, to be able to find the exact values of the corresponding image, we need to know the exact values of the directly measured quantities. In reality, measurements are never 100% accurate, the measured value $\widetilde{x}$ is usually somewhat different from the (unknown) actual value x of the measured quantity. In other words, in practice, there is a non-zero measurement error $\Delta x \stackrel{\text{def}}{=} \widetilde{x} - x$.

Our numerical methods process the measured values $\widetilde{x}$ – which serve as the input data to the corresponding numerical algorithms. Since these values are, in general, different from the actual value x, the resulting solution is different from the actual image. It is therefore desirable to estimate the difference between the numerical solution and the actual image which is caused by the uncertainty with which we know the inputs.

The analysis of this effect is sometimes called *sensitivity analysis.* This analysis is what we will do in this chapter.

4.3 Input Uncertainty and How It Affects the Results of Data Processing: A Brief Reminder

It is important to have some information about the input uncertainty. In order to gauge the effect of the input uncertainty on the difference between the numerical solution and the actual image, we must have information about the input uncertainty, i.e., the information about the measurement errors $\Delta x = \widetilde{x} - x$ of the corresponding (direct) measurements.

Upper bound on the measurement error. How can this measurement error be described? First, the manufacturer of a measuring instrument must provide us with an upper bound Δ on the absolute value $|\Delta x|$ of the measurement error Δx. If no such bound was guaranteed, this would mean that the difference Δx can be arbitrarily large; in this situation, after getting a measurement result, say, $\widetilde{x} = 1$, we cannot be sure whether the actual value x of the measured quantity is 1, 0, 10, 100, or 1,000,000. In this situation, $\widetilde{x} = 1$ is a wild guess, not a measurement result.

When we know this upper bound Δ, this means that the actual value Δx of the measurement error must be inside the interval $[-\Delta, \Delta]$.

Probabilistic information. In addition to the upper bound Δ, we often also know the probabilities of different values Δx from the interval $[-\Delta, \Delta]$.

This situation of probabilistic uncertainty is traditionally used in engineering and scientific practice, especially the case when the measurement error is normally distributed. In particular, there exist well-developed method for determining how this probabilistic uncertainty in the inputs affects the difference between the numerical solution and the actual image.

Case when probabilistic information is not readily available. In many important practical situations, we do not have the information about the probabilities of different values of Δx, we only know the upper bound Δ.

The reason is that the probabilistic information usually comes from comparing the results of measuring the same quantity with two different measuring instruments: the one used for actual measurements and the *standard* (much more accurate) one – whose results are so much closer to the actual values that we can ignore the corresponding measurement errors and consider these results actual values.

There are two situations when this comparison is not done. The first such situation is the situation of cutting-edge measurements, when we are actually using the best possible

measuring instrument. For example, if we perform some protein measurements by using a state-of-the-art electronic microscope, it would be nice to be able to compare the results with a much more accurate microscope – but ours is already the best.

Another case when the probabilities are not determined is when we have limited resources. For example, in geophysics, in every seismic experiment, we use a large number of sensors to measure the corresponding travel times. It would be nice to be able to compare all these sensors with more accurate ones. However, the detailed comparison of each sensor requires the use of costly standard sensors and, as a result, costs several orders of magnitude more than the cost of buying a new sensor – so we often cannot do this detailed probabilistic "calibration" within our limited resources.

In both cases, the only information we have about the measurement error $\Delta x = \widetilde{x} - x$ is the upper bound Δ: $|\Delta x| \leq \Delta$. In such situations, once we have the measurement result $\widetilde{x}$, the only conclusion that we can make about the (unknown) actual value x is that x belongs to the interval $\mathbf{x}_i = [\underline{x}_i, \overline{x}_i]$, where $\underline{x}_i \stackrel{\text{def}}{=} \widetilde{x} - \Delta$ and $\overline{x}_i \stackrel{\text{def}}{=} \widetilde{x} + \Delta$. This situation is called the situation of *interval uncertainty*.

In this chapter, we will concentrate on interval uncertainty. While, as we have just mentioned, interval uncertainty is practically important, the analysis of this uncertainty is much less developed that the analysis of the probabilistic uncertainty. Because of this fact, in this chapter, we will concentrate on the analysis of how interval input uncertainty affects the difference between the numerical solution and the actual image.

Interval computations. Let us describe the general situation of data processing under interval uncertainty in precise terms. We have an algorithm $f(x_1, \ldots, x_n)$ which transforms the values $x_1, \ldots, x_n$ of directly measurable quantities into the value $y = f(x_1, \ldots, x_n)$ of the desired quantity y. We do not know the actual values x_i; instead, we only know the intervals $[\underline{x}_i, \overline{x}_i]$ of possible values of x_i.

Different values $x_i \in [\underline{x}_i, \overline{x}_i]$ lead, in general, to different values of $y = f(x_1, \ldots, x_n)$. It

is therefore desirable to find the range $\mathbf{y}$ of possible values of y:

$$\mathbf{y} = \{f(x_1, \ldots, x_n) \,|\, x_1 \in [\underline{x}_1, \overline{x}_1], \ldots, x_n \in [\underline{x}_n, \overline{x}_n]\}.$$

The function $f(x_1, \ldots, x_n)$ computed by this algorithm is usually continuous. It is known that the range of a continuous function $f(x_1, \ldots, x_n)$ over a connected bounded closed set $[\underline{x}_1, \overline{x}_1] \times \ldots \times [\underline{x}_n, \overline{x}_n]$ is a closed interval. So, the desired range $\mathbf{y}$ is an interval: $\mathbf{y} = [\underline{y}, \overline{y}]$ for some values $\underline{y}$ and $\overline{y}$. Thus, to compute the range $\mathbf{y}$ means to compute the endpoints $\underline{y}$ and $\overline{y}$ of the corresponding interval. The problem of computing these endpoints is known as the problem of *interval computations*; see, e.g., [94].

Monotonicity: simplest case of interval computations. The simplest case in which we can estimate the range of a function $f(x_1, \ldots, x_n)$ over a given box $[\underline{x}_1, \overline{x}_1] \times \ldots \times [\underline{x}_n, \overline{x}_n]$ is the case when the function $f(x_1, \ldots, x_n)$ is monotonic in each of the variables.

For example, when the function $f(x_1, \ldots, x_n)$ is increasing in each of the variables x_i, then its value is the smallest when all the inputs x_i attain their smallest values $x_i = \underline{x}_i$, and its value is the largest when all the inputs x_i attain their largest values $x_i = \overline{x}_i$. In other words, in this case, $\mathbf{y} = [\underline{y}, \overline{y}]$, where $\underline{y} = f(\underline{x}_1, \ldots, \underline{x}_n)$ and $\overline{y} = f(\overline{x}_1, \ldots, \overline{x}_n)$.

Similar formulas can be written when the function $f(x_1, \ldots, x_n)$ is decreasing in all of its variables, or when it is increasing in some of the variables and decreasing in some other variables. For example, if a function $f(x_1, x_2)$ is increasing in x_1 and decreasing in x_2, then its range on the box $[\underline{x}_1, \overline{x}_1] \times [\underline{x}_2, \overline{x}_2]$ is equal to $[\underline{y}, \overline{y}] = [f(\underline{x}_1, \overline{x}_2), f(\overline{x}_1, \underline{x}_2)]$.

How typical is a monotonic case? According to calculus, a function f is increasing in x_i if its partial derivative is non-negative: $\dfrac{\partial f}{\partial x_i} \geq 0$. Similarly, a function f is decreasing in x_i if its partial derivative is non-positive: $\dfrac{\partial f}{\partial x_i} \leq 0$. So, to make sure that a function f is either increasing or decreasing on a box, it is sufficient to make sure that its partial derivative stays non-negative or stays non-positive, and never crosses 0.

For most functions $f(x_1, \ldots, x_n)$, the equation $\dfrac{\partial f}{\partial x_i} = 0$ defines an $(n-1)$-dimensional surface in the n-dimensional space. In practice, the actual values of $x_1, \ldots, x_n$ are randomly

distributed in the n-dimensional space. Thus, it is reasonable to expect that the probability of having the partial derivative to be 0 is equal to 0. In other words, with confidence 1, we can conclude that the values of all the partial derivatives are non-zeros.

We want the derivative to be not crossing 0 not only at the point $x = (x_1, \ldots, x_n)$ corresponding to the actual (unknown) values of the input quantities $x_1, \ldots, x_n$, but also for all the values for which $x_i \in [\underline{x}_i, \overline{x}_i]$. Since the derivative is usually continuous, when the values x_i are close, the values of the derivative are also close. Thus, if the measurements are accurate enough – and the corresponding intervals $[\underline{x}_i, \overline{x}_i]$ are narrow enough – each of n partial derivatives $\dfrac{\partial f}{\partial x_i}$ retains the same sign throughout the entire box – the same sign as it has for the actual values.

So, if the measurements are accurate enough, we can use monotonicity to compute the range of the function $f(x_1, \ldots, x_n)$.

For wider input intervals (corresponding to slightly less accurate measurements), a reasonable idea is to divide the original box into narrower subboxes; on most of these subboxes, the function is monotonic, so interval computations are straightforward; see, e.g., [94].

Interval computations in the general non-monotonic case. How can we estimate the range of a function $f(x_1, \ldots, x_n)$ on a box on which this function is not monotonic? In general, the problem of computing this range is known to be NP-hard. This means, crudely speaking, that we cannot expect an efficient algorithm that would always compute the exact range.

There are, however, reasonable algorithms which provide the *enclosure* $\mathbf{Y} \supseteq \mathbf{y}$ for the desired range. Historically, the first approach to such an estimation is based on the so-called *straightforward interval computations* techniques. The main idea behind these techniques is that elementary arithmetic operations (addition, subtraction, multiplication, division) are monotonic and thus, for these operations, the range is easy to compute. For example, the addition function $f(x_1, x_2) = x_1 + x_2$ is increasing in both variables, so its range can

be computed as

$$f([\underline{x}_1, \overline{x}_1], [\underline{x}_2, \overline{x}_2]) = [\underline{x}_1 + \underline{x}_2, \overline{x}_1 + \overline{x}_2].$$

Since the function $f(x_1, x_2) = x_1 + x_2$ is usually described by using an infix notation – with the function symbol between the inputs, it is reasonable to denote this range by using infix notations too, as $[\underline{x}_1, \overline{x}_1] + [\underline{x}_2, \overline{x}_2]$. In these notations, the above formula for the range takes the form

$$[\underline{x}_1, \overline{x}_1] + [\underline{x}_2, \overline{x}_2] = [\underline{x}_1 + \underline{x}_2, \overline{x}_1 + \overline{x}_2].$$

Similarly, subtraction $f(x_1, x_2) = x_1 - x_2$ is increasing in x_1 and decreasing in x_2; so, in the infix notations, the formula for the range of the subtraction takes the following form:

$$[\underline{x}_1, \overline{x}_1] - [\underline{x}_2, \overline{x}_2] = [\underline{x}_1 - \overline{x}_2, \overline{x}_1 - \underline{x}_2].$$

For multiplication $f(x_1, x_2) = x_1 \cdot x_2$, the direction of monotonicity depends on the signs of x_1 and x_2. In all cases, however, whether the function is increasing or decreasing in each of the variables, both its minimum and its maximum are attained at some combination of endpoints. Thus, we can conclude that

$$[\underline{x}_1, \overline{x}_1] \cdot [\underline{x}_2, \overline{x}_2] = [\min(\underline{x}_1 \cdot \underline{x}_2, \underline{x}_1 \cdot \overline{x}_2, \overline{x}_1 \cdot \underline{x}_2, \overline{x}_1 \cdot \overline{x}_2), \max(\underline{x}_1 \cdot \underline{x}_2, \underline{x}_1 \cdot \overline{x}_2, \overline{x}_1 \cdot \underline{x}_2, \overline{x}_1 \cdot \overline{x}_2)].$$

For division $f(x_1, x_2) = x_1/x_2$, similar formulas hold but we need to make sure that 0 is not in the range of possible values of x_2: if $0 \notin [\underline{x}_2, \overline{x}_2]$, then

$$[\underline{x}_1, \overline{x}_1]/[\underline{x}_2, \overline{x}_2] = [\min(\underline{x}_1/\underline{x}_2, \underline{x}_1/\overline{x}_2, \overline{x}_1/\underline{x}_2, \overline{x}_1/\overline{x}_2), \max(\underline{x}_1/\underline{x}_2, \underline{x}_1/\overline{x}_2, \overline{x}_1/\underline{x}_2, \overline{x}_1/\overline{x}_2)].$$

These formulas are called formulas of *interval arithmetic*.

For functions $f(x_1, \ldots, x_n)$ which are more complex than simple arithmetic operations, we can take into account that inside the computer, each algorithm is translated ("parsed") into a sequence of elementary (arithmetic) operations. It turns out that if we replace each operation with numbers with the corresponding operation of interval arithmetic, then at the end, we get an enclosure for the desired range. This technique is called *straightforward interval computations*.

There exist more efficient techniques for computing the interval range, e.g., techniques based on the centered form [94]. In a nutshell, however, these techniques are also based on the use of straightforward interval computations. The main innovation is that, crudely speaking, the straightforward interval computations are applied not to the original parsing of an algorithm but to a parsing which is designed to improve the resulting enclosures.

How do we check monotonicity? As we have mentioned, if a function $f(x_1, \ldots, x_n)$ is monotonic on the whole box, then the corresponding interval computations problem is easy to solve. If it is not monotonic on the entire box, then for each subbox for which it is monotonic, the computations of the corresponding range become much faster. It is therefore important to be able to check whether a given function is monotonic on a given box.

Since monotonicity means the constant sign of the corresponding partial derivative, a natural way to check monotonicity is to compute the range of this partial derivative, and to make sure that this range does not cross 0. For computing range, as we have mentioned, we can use interval computations techniques. Thus, a natural way to check monotonicity is to use interval computation techniques.

What we will do. In this chapter, we will use interval computations to check monotonicity for a simple case study.

4.4 Case Study: A Description

Case of images: reminder. We are interested in the situation when the desired values form a 2-D or 3-D image, i.e., they correspond to the values $u(\vec{x})$ of a certain quantity (such as intensity or velocity of sound) at different points $\vec{x}$ in the 2-D or 3-D space.

In this chapter, we consider the situation when we cannot measure the values $u(\vec{x})$ directly, we can only measure the values $f(\vec{x})$ of some other field which is related to u by a known relation.

In more precise terms, we have a mapping M which maps functions into functions; such mappings are usually called *operators*. In terms of this known mapping (operator), the relation between the unknown image $u(\vec{x})$ and the directly measurable (known) image $I(\vec{x})$ can be described as $M(u) = I$. In this setting, we must determine u from the equation $M(u) = I$.

Linear operators: an important practical case. In general, the dependence of I on u can be non-linear. In many practical situations, however, the values of the unknown image $u(\vec{x})$ are reasonably small, so that we can safely ignore terms which are quadratic or higher order in terms of u.

In the discrete case, when we only consider finitely many points $\vec{x}$, if we expand the dependence M of each value $I(\vec{x})$ on the values $u(\vec{y})$ in Taylor series and ignore all quadratic and higher-order terms, we end up with a linear dependence

$$I(\vec{x}) = I_0(\vec{x}) + \sum_{\vec{y}} L(\vec{x}, \vec{y}) \cdot u(\vec{y})$$

for some coefficients $I_0(\vec{x})$ and $L(\vec{x}, \vec{y})$.

It is usually convenient to keep all the terms depending on the unknown on one side of the linear equation, and move all other terms to other side. By moving the term $I_0(\vec{x})$ to the other side, we obtain a linear equation

$$\sum_{\vec{y}} L(\vec{x}, \vec{y}) \cdot u(\vec{y}) = I(\vec{x}) - I_0(\vec{x}).$$

We know the values $I(\vec{x})$ from measurements; we can determine the values $I_0(\vec{x})$ by plugging in the zero function $u(\vec{x}) = 0$ into the operator M. Thus, we can determine the difference $f(\vec{x}) \stackrel{\text{def}}{=} I(\vec{x}) - I_0(\vec{x})$ in the right-hand side of the above equation. We can therefore simplify the description of the problem by saying that we need to determine the values $u(\vec{y})$ from the system of linear equations

$$\sum_{\vec{y}} L(\vec{x}, \vec{y}) \cdot u(\vec{y}) = f(\vec{x}),$$

in which we know the right-hand side $f(\vec{x})$ and the coefficients $L(\vec{x}, \vec{y})$.

In the continuous case, we have a similar linearized problem: we know the function $f(\vec{x})$, we know a linear operator $L(u)$, and we must find the function $u(\vec{x})$ which satisfies the equation $L(u) = f$.

Case of interval uncertainty. As we have mentioned earlier, in practice, after measurements, we often do not get the exact values $f(\vec{x})$. Instead, for each $\vec{x}$, we get the interval $[\underline{f}(\vec{x}), \overline{f}(\vec{x})]$ of possible values of $f(\vec{x})$.

Different functions $f(\vec{x})$ which satisfy the interval constraints $f(\vec{x}) \in [\underline{f}(\vec{x}), \overline{f}(\vec{x})]$ lead, in general, to different functions $u(\vec{x})$.

It is therefore desirable to find, for each $\vec{x}$, the interval $[\underline{u}(\vec{x}), \overline{u}(\vec{x})]$ of possible values of $u(\vec{x})$.

Computing the range is easier in the monotonic case. The problem of computing the interval $[\underline{u}(\vec{x}), \overline{u}(\vec{x})]$ becomes much simpler if the "inverse" mapping L^{-1} that maps each function f to the solution u of the equation $L(u) = f$ is monotonic in the following sense: if $f_1(\vec{x}) \le f_2(\vec{x})$ for all $\vec{x}$, then the corresponding solutions, i.e., the functions u_1 and u_2 which satisfy the equations $L(u_1) = f_1$ and $L(u_2) = f_2$, satisfy the inequality $u_1(\vec{x}) \le u_2(\vec{x})$ for all $\vec{x}$.

In this case, the desired bounds for $u(\vec{x})$ can be found easily. Indeed, we know that for every $\vec{x}$, we have $\underline{f}(\vec{x}) \le f(\vec{x}) \le \overline{f}(\vec{x})$. Thus, due to the monotonicity of the inverse mapping, we have $\underline{u}(\vec{x}) \le u(\vec{x}) \le \overline{u}(\vec{x})$, where:

- $\underline{u}$ is the solution to the equation $L(\underline{u}) = \underline{f}$;
- u is the solution to the equation $L(u) = f$; and
- $\overline{u}$ is the solution to the equation $L(\overline{u}) = \overline{f}$.

Thus, for every $\vec{x}$ and for each of these solutions, we get $u(\vec{x}) \in [\underline{u}(\vec{x}), \overline{u}(\vec{x})]$. Since both endpoints $\underline{u}(\vec{x})$ and $\overline{u}(\vec{x})$ are possible values of $u(\vec{x})$ – they correspond to the functions

$\underline{f}(\vec{x}) \in [\underline{f}(\vec{x}), \overline{f}(\vec{x})]$ and $\overline{f}(\vec{x}) \in [\underline{f}(\vec{x}), \overline{f}(\vec{x})]$ – the interval of possible values of $u(\vec{x})$ is exactly the interval $[\underline{u}(\vec{x}), \overline{u}(\vec{x})]$ formed by the functions $\underline{u} = L^{-1}(\underline{f})$ and $\overline{u} = L^{-1}(\overline{f})$.

Terminological comment: how to avoid confusion between different notions of monotonicity. Before we start checking monotonicity of different mappings, let us first clarify the terminology.

In our analysis, we use the notion of monotonicity which comes from calculus, from analyzing the properties of general functions. In general, monotonicity is a very straightforward notion which should not cause any confusion.

However, in the linear case, there is a potential confusion. This potential confusion is caused by the fact that in the linear case, the mapping L^{-1} from functions to functions is a linear operator. For linear operators on a Hilbert space, e.g., for mappings defined on the set H of real-valued square integrable functions $u(\vec{x})$ with $\int u(\vec{x})^2\,d\vec{x} < +\infty$, there is a different notion of monotonicity: namely, an operator A from a Hilbert space to itself is called monotonic if $\langle Au, u\rangle \geq 0$ for all elements u, where for the space H, we have $\langle u, v\rangle \stackrel{\text{def}}{=} \int f(\vec{x}) \cdot g(\vec{x})\,d\vec{x}$.

This condition is, in general, different from the notion of monotonicity in the calculus sense. For example, if we consider a mapping $Au(x) = -u''$ which translates every function $u(x)$ of one variable into its second derivative, and limit this mapping to functions defined on some interval $[a, b]$ which are equal to 0 at the endpoints of this interval, then we get

$$\langle Au, u\rangle = \int_a^b (Au)(x) \cdot u(x)\,dx = -\int_a^b u''(x) \cdot u(x)\,dx.$$

Integrating by parts and taking into account that $u(a) = u(b) = 0$, we conclude that

$$\langle Au, u\rangle = \int_a^b (u'(x))^2\,dx$$

and thus, that $\langle Au, u\rangle \geq 0$ – i.e., that this operator A is monotonic in the operator sense.

On the other hand, it is easy to show that this operator is not monotonic in the calculus sense. For example, a function $u_1(x) = 0$ is defined on the interval $[0, 2\pi]$ and is equal

to 0 at both endpoints of this interval. Similarly, the function $u_2(x) = 1 - \cos(x)$ is defined for all $x \in [0, 2\pi]$ and is also equal to 0 at both endpoints of this interval. It is easy to check that $u_1(x) \le u_2(x)$ for all $x \in [0, 2\pi]$. If the operator $Au = -u''$ was monotonic in the calculus sense, then we would have $-u_1''(x) \le -u_2''(x)$ for all x. For these two functions, however, $-u_1''(x) = 0$ and $-u_2''(x) = \cos(x)$, so, e.g., for $x = \pi$, we have $-u_1''(\pi) = 0 > -u''(\pi) = \cos(\pi) = -1$.

What is the proper linear operator description of the calculus-motivated notion of monotonicity? To answer this question, let us use the fact that in the above example, it was sufficient to consider the cases when $u_1(x) = 0$. When restricted to such cases, the calculus-motivated monotonicity property takes the following form: if $0 \le u(\vec{x})$ for all $\vec{x}$, then $0 \le (Au)(\vec{x})$ for all $\vec{x}$. Vice versa, if we have this property for all functions u, then whenever we have $u_1(\vec{x}) \le u_2(\vec{x})$ for all $\vec{x}$, we have $\Delta u(\vec{x}) \ge 0$, where $\Delta u(\vec{x}) \stackrel{\text{def}}{=} u_2(\vec{x}) - u_1(\vec{x})$. Thus, we have $(A\Delta u)(\vec{x}) \ge 0$. Since A is a linear operator, $A(\Delta u) = Au_2 - Au_1$, so $(A(\Delta u))(\vec{x}) \ge 0$ means that $(Au_1)(\vec{x}) \le (Au_2)(\vec{x})$ – i.e., monotonicity in the calculus-motivated sense.

So, in terms of linear operators A, calculus based monotonicity means that if $u \ge 0$, then $Au \ge 0$. In other words, the operator A must transform non-negative functions into non-negative ones – i.e., *preserve* non-negativity. This is exactly how this property is called in the analysis of linear operators – *non-negativity preservation*.

So, for linear operators, monotonicity in the calculus-motivated sense means non-negativity preservation. In these terms, we are looking for situations in which the inverse operator L^{-1} is non-negativity preserving.

Simple example of a non-negativity-preserving inverse operator. A simple example of a non-negativity-preserving inverse operator is given by the solution to the following simple differential equation

$$-u'' = f \tag{4.1}$$

on a given open interval, e.g., on an interval $\Omega = (-1, 1)$, with the boundary condition that the solution $u(x)$ is equal to 0 at the endpoints of this interval: $u(-1) = u(1) = 0$.

Non-negativity means that if the right-hand side $f(x)$ is everywhere non-negative, i.e., $f(x) \geq 0$ for all x, then the solution $u(x)$ is also non-negative. For this simple equation, this is indeed the case: if $f(x) \geq 0$, this means that $u''(x) \leq 0$ for all x, i.e., that the function $u(x)$ is concave, i.e., $u(\alpha x + (1-\alpha) \cdot y) \geq \alpha \cdot u(x) + (1-\alpha) \cdot u(y)$. Since $u(-1) = u(1) = 0$, we can conclude that $u(x) \geq 0$ for all $x \in (-1, 1)$. Thus, the above inverse operator is indeed non-negativity preserving.

Comment. It is worth mentioning that the operator $-u''$ is elliptic, and that the above non-negativity preservation property is a particular case of the maximum principle for elliptic problems.

The need for approximations. In the above example, we (implicitly) assumed that we can have an arbitrary function $f(x)$. To describe an arbitrary function $f(x)$, we need infinitely many parameters – e.g., the values of this function at different points x. A computer program can only process a finite amount of bits, so in the computer, we can only represent finitely many parameter values. Thus, in practice, we must consider the case when the right-hand side $f(x)$ is described by a finite-parametric family of functions. Similarly, instead of the exact solution $u(x)$ to the above equation, we only look for functions from the similar finite-parametric family.

Polynomial and piece-wise polynomial approximations. In principle, we can select different approximating families. Since our objective is computations, between different families, it is reasonable to select families of functions which are the easiest to compute. In the computer, the fastest operations with real numbers are directly hardware-supported arithmetic operations, i.e., addition, subtraction, and multiplication. It is therefore reasonable to restrict ourselves to functions which are obtained from the unknowns and constant by using addition, subtraction, and multiplication. These functions are exactly *polynomials*,

so a natural idea is to use polynomial approximations to the actual functions.

This is how many elementary functions such as $\sin(x)$, $\cos(x)$, $\exp(x)$, etc., are implemented in most computer systems – by using appropriate polynomial approximations to these functions.

Finite element method (FEM). It is worth mentioning that for some of these elementary functions, the computer implementations use different approximating polynomials on different parts of the input range. So, in general, we can use piece-wise polynomial functions composed of different polynomial pieces. Such piece-wise polynomial approximations are the basis of one of the most widely used techniques for solving differential equations – the Finite Element Method (FEM); see, e.g., [198, 200]. In this method, the domain is divided into "elements" (usually, polyhedra), and each desired function is described, on each element, by a polynomial of a given degree. Traditionally, linear functions are used on each element, but P. Šolín and others have successfully shown that in many practical situations, the use of higher-order approximating polynomials is extremely beneficial – it leads to higher accuracy for the same computations or, equivalently, to a drastic decrease of computation time which is needed to achieve a given accuracy [198, 200].

FEM notations. The corresponding approximations to the functions $f(x)$ and $u(x)$ are usually denoted by $f_{h,p}(x)$ and $u_{h,p}(x)$, where h is the size of the element, and p is the order of the corresponding approximating polynomials.

The class of all possible right-hand sides (i.e., the class of all piece-wise polynomial functions) is usually denoted by $W_{h,p}$, and the class of all possible solutions – i.e., all piece-wise polynomial functions which satisfy the corresponding boundary conditions – is usually denoted by $V_{h,p}$. By this definition, the class $V_{h,p}$ is a subclass of the class $W_{h,p}$: $V_{h,p} \subseteq W_{h,p}$.

Restriction to polynomial approximations. In this section, we want to explain the use of interval techniques on the simplest possible (non-trivial) example. Because of this

need for simplicity, we will restrict ourselves to the simplest case – of polynomial approximations.

Comment. In FEM terms, this restriction means that we consider the cases when the whole domain is used as a single element. For traditional FEM, when we approximate a function on each element by a linear expression, this restriction may sound too crude. However, in higher-order FEM methods, when we use approximating polynomials of higher order, this restriction is actually not that restrictive: it is well known that an arbitrary continuous function can be, for any given approximation accuracy, approximated by a polynomial (of a sufficiently high order).

Towards a finite element formulation of the original problem. In the FEM, we assume that function $f_{h,p}(x)$ belongs to the class $W_{h,p} = P^p(-1,1)$ of all polynomials of p-th order defined on the interval $(-1,1)$. We are looking for a function $u_{h,p}(x)$ from the class $V_{h,p} = P_0^p(-1,1)$ of all the polynomials of p-th order which vanish at the endpoints $x = -1$ and $x = 1$ of this interval.

At first glance, it may sound reasonable to consider the same equation like $-u'' = f$ for the polynomial approximations as well. In other words, it may sound reasonable to consider the equation $-u''_{h,p}(x) = f_{h,p}(x)$, or, equivalently, $u''_{h,p}(x) + f_{h,p}(x) = 0$ for all $x \in (-1,1)$. In reality, this is not possible. For example, if we follow the traditional FEM approach and consider linear approximations $u_{h,p}(x)$ to u on each element, then the second derivative $u''_{h,p}(x)$ is always 0 – as a second derivative of a linear function, and thus, the equation $-u''_{h,p}(x) = f_{h,p}(x)$ cannot be satisfied for $f_{h,p}(x) \neq 0$.

This example shows that we cannot require that the value $z(x) \stackrel{\text{def}}{=} u''_{h,p}(x) + f_{h,p}(x)$ be equal to 0 for all $x \in (-1,1)$. We still want to claim that within this approximation, the auxiliary function $z(x)$ is, in some reasonable sense, indistinguishable from 0.

Let us recall that in reality, we rarely observe the actual value $I(\vec{x})$ of the image I at a single point $\vec{x}$; usually, every measuring instrument covers a region around the point, and

the observed value is the weighted average over this region, i.e., the value $\int I(\vec{x}) \cdot w(\vec{x})\, d\vec{x}$ for some weight function $w(\vec{x})$. So, to claim that the function $z(x)$ is indistinguishable from 0 means that the corresponding "observable" integral $\int z(x) \cdot w(x)\, dx$ is equal to 0 for all appropriate weight functions $w(x)$. Since we restricted all the approximating functions to polynomials of given order p, it is reasonable to consider only polynomials of this order as appropriate weight functions $w(x)$. Since we are only interested in the values x from the open interval $(-1, 1)$, we should require that the weight function turns into 0 at the endpoints of this interval.

Thus, we arrive at the following finite element reformulation of the original problem:

$$\int_\Omega z(x) \cdot w_{h,p}(x)\, dx = \int_\Omega (u''_{h,p}(x) + f_{h,p}(x)) \cdot w_{h,p}(x)\, dx = 0$$

for all the functions $w_{h,p}$ from the space $V_{h,p} = P_0^p(-1, 1)$ of all polynomials of order p that vanish at the endpoints $x = -1$ and $x = 1$. Integrating the integral $\int_\Omega u''_{h,p}(x) \cdot w_{h,p}(x)\, dx$ by parts and taking into account that $w_{h,p}(-1) = w_{h,p}(1) = 0$, we conclude that

$$\int_\Omega u'_{h,p}(x) \cdot w'_{h,p}(x)\, dx = \int_\Omega f_{h,p}(x) \cdot w_{h,p}(x)\, dx \text{ for all } w_{h,p} \in V_{h,p}. \tag{4.2}$$

This is the desired finite-element analog of the original differential equation.

How to get from an arbitrary function f to its finite-parametric approximation $f_{h,p}$. A similar idea can be used to describe how to transform the original right-hand side $f(x)$ to the finite-parametric approximation $f_{h,p}(x)$. Indeed, ideally, we would like to have $f(x) = f_{h,p}(x)$ for all x, i.e., $z(x) \stackrel{\text{def}}{=} f(x) - f_{h,p}(x) = 0$. Since we cannot have $z(x) = 0$ for all x, we must therefore require that all corresponding "observable" quantities $z(x) \cdot w(x) = 0$, i.e., that

$$\int_\Omega z(x) \cdot w_{h,p}(x)\, dx = \int_\Omega (f(x) - f_{h,p}(x)) \cdot w_{h,p}(x)\, dx = 0 \text{ for all } w_{h,p} \in W_{h,p}.$$

The operation $\int f(x) \cdot g(x)\, dx$ is a dot-product between the two functions $f(x)$ and $g(x)$; this interpretation is a continuous analog of the dot-product $a \cdot b = a_1 \cdot b_1 + \ldots + a_i \cdot b_i + \ldots + a_n \cdot b_n$

between two vectors $a = (a_1, \ldots, a_i, \ldots, a_n)$ and $b = (b_1, \ldots, b_i, \ldots, b_n)$, an analog in which we have a continuous variable x instead of a discrete variable i. Similar to the case of the two vectors, two functions for which the dot-product are 0 are called *orthogonal* to each other.

In these terms, we represent the original function $f(x)$ as a sum of the two terms: the term $f_{h,p}(x) \in W_{h,p}$ and the term $f(x) - f_{h,p}(x)$ which is orthogonal to all the functions from $W_{h,p}$. For vectors, this is called *projection*; so we can say that the corresponding finite-parametric function $f_{h,p}(x)$ is a *projection* of the original function $f(x) \geq 0$ on the finite-parametric space $W_{h,p}$.

Does the non-negativity preservation property hold for this finite-parametric approximation? We have already mentioned that the original inverse operator is non-negativity preserving in the following precise sense: If the right-hand side is non-negative, i.e., if $f(x) \geq 0$ for all x, then the solution $u(x)$ is also non-negative, i.e., $u(x) \geq 0$ for all x.

It is reasonable to expect that the same will be true for the finite-parametric approximation to the inverse operator, i.e., that if the original right-hand side is non-negative ($f(x) \geq 0$ for all x), then:

- the corresponding projection $f_{h,p}(x)$ is also non-negative, i.e., $f_{h,p}(x) \geq 0$ for all x, and
- the resulting solution $u_{h,p}(x)$ should also be non-negative, i.e., $u_{h,p}(x) \geq 0$ for all x.

Results of this type are called *discrete nonnegativity conservation principles (DNCP)*. Such results are known for some operators. However, for many linear operators, it is still an open problem whether DNCP holds – because DNCP results are very difficult to prove even for the simplest PDEs discretized by the simplest numerical methods [43, 44].

DNCP results are known for various types of linear-order methods, such as for piecewise-affine FEM; see, e.g., [96, 119, 121, 221, 222]. For higher-order finite element methods,

methods which are often more efficient (see, e.g., [198] and the references therein), virtually no DNCP results have been available until our recent paper [202].

Counterexample. In the early 1980s, Höhn and Mittelmann [90] have shown that for higher-order FEM, DNCP (as formulated above) does not hold even for the simplest equations. This negative result can be illustrated on the above example of the equation $-u'' = f$ on the interval $(-1, 1)$. For the non-negative right-hand side

$$f(x) = 200\mathrm{e}^{-10(x+1)}, \tag{4.3}$$

with $p = 3$, the resulting third order finite-element solution

$$u_{h,p}(x) = \frac{1}{40}\left[54 + 66\mathrm{e}^{-20} - (73 - 133\mathrm{e}^{-20})x\right]\left(1 - x^2\right) \tag{4.4}$$

is negative for some points $x \in \Omega$ – e.g., at $x = 0.9$.

Analysis of the counterexample. A detailed analysis of this example shows that in this case, already the approximating function $f_{h,p}(x)$ is negative at some points x. So, a natural question is: what if we only consider functions for which $f_{h,p}(x)$ is non-negative? Will then the non-negativity preserving property hold, i.e., will then the solution $u_{h,p}(x)$ also be non-negative?

New formulation of DNCP. This natural question leads to a new formulation of DNCP: if $f_{h,p}(x) \geq 0$, then $u_{h,p}(x) \geq 0$.

What we prove. In this chapter, we use interval computations to prove that for the equation $-u'' = f$ and for realistic values p (specifically, for $p \leq 10$), the above DNCP property indeed holds.

4.5 Proof of Non-Negativity Preservation of the Simple Inverse Operator: Main Ideas and the Need for Interval Computations

From a standard representation of a polynomial to Legendre polynomials. In our problem, the right-hand side $f_{h,p}(x)$ and the desired solution $u_{h,p}(x)$ are polynomials of p-th order. To describe each polynomial, we need finitely many parameters.

One possible way to describe a general polynomial of p-th order $f(x) = a_0 + a_1 \cdot x + \ldots + a_p \cdot x^p$ is to describe it by its coefficients $a_0, a_1, \ldots, a_p$. In linear algebra terms, this means that we select a basis consisting of monomials 1, x, x^2, x^3, $\ldots$, and we represent each polynomial by the coefficients of its representation as a linear combination of different functions from this basis.

The problem with this seemingly natural choice of a basis is that the most convenient-to-use bases are orthogonal bases – especially since some approximation properties are naturally formulated in terms of orthogonality – and the monomials are not orthogonal on the interval $(-1, 1)$. It is therefore reasonable to use instead a basis which is orthogonal on the interval $(-1, 1)$. This basis can be formed by a standard orthogonalization procedure applied to the monomials. Specifically, we start with the monomial $L_0(x) = 1$. It so happened that x is orthogonal to 1 on the interval $(-1, 1)$, so we take $L_1(x) = x$. The monomial x^2 is not orthogonal to 1 and x, so instead of x^2, we take a linear combination of 1, x, and x^2 which is orthogonal to x. If we additionally require that $L_2(1) = 1$, we get $L_2(x) = \dfrac{1}{2} \cdot (3x^2 - 1)$. The resulting orthogonal polynomials are known as *Legendre polynomials*.

Comment. Similar to the vector's length (norm) $\|a\| = \sqrt{a \cdot a} = \sqrt{a_1^2 + \ldots + a_n^2}$, we can define the norm of a function as $\|f\|_{L^2} = \sqrt{\int_{-1}^{1} f(t)^2\, dt}$. It is known that the norm of the Legendre polynomial is equal to $\|L_k\|_{L^2} = \sqrt{2/(2k+1)}$.

From Legendre polynomials to Lobatto shape functions. Since the Legendre polynomials are orthogonal to each other on the interval $(-1, 1)$, all Legendre polynomials $L_k(x)$ of order $k \geq 1$ are orthogonal to $L_0(x) = 1$, i.e., $\int_{-1}^{1} L_k(x)\,dx = 0$. So, for each k, the integral function $F(x) = \int_{-1}^{x} L_k(t)\,dt$ is a polynomial of order $k+1$ which has the property that $F(-1) = F(1) = 0$.

Since we are interested in solutions for which $u(-1) = u(1) = 0$, it is reasonable to use these integral functions as a basis for our solutions. For convenience, let us "normalize" the Legendre polynomials so that their norm is equal to 1. The integrals of such normalized Legendre polynomials are known as *Lobatto shape functions* (see, e.g., [200]):

$$l_k(x) = \frac{1}{\|L_{k-1}\|} \int_{-1}^{x} L_{k-1}(\xi)\ \mathrm{d}\xi, \quad 2 \leq k, \tag{4.5}$$

where

$$\|L_i\| \stackrel{\text{def}}{=} \sqrt{\int_{-1}^{1} L_i(x)^2\,dx}.$$

Each Lobatto shape function $l_k(x)$ is a polynomial of the corresponding order. Since this function is equal to 0 for $x = -1$ and for $x = 1$, the corresponding polynomial divides $x - (-1) = x + 1$ and $x - 1$ hence, divides $x^2 - 1$, i.e., $l_k(x) = (x^2 - 1) \cdot \widetilde{l}_k(x)$ for some polynomial $\widetilde{l}_k(x)$. Here are the explicit expressions for the functions $\widetilde{l}_k(x)$ for $k = 2, \ldots, 10$:

$$\widetilde{l}_2(x) = \frac{1}{2} \cdot \sqrt{\frac{3}{2}}; \quad \widetilde{l}_3(x) = \frac{1}{2} \cdot \sqrt{\frac{5}{2}} \cdot x; \quad \widetilde{l}_4(x) = \frac{1}{8} \cdot \sqrt{\frac{7}{2}} \cdot (5x^2 - 1);$$

$$\widetilde{l}_5(x) = \frac{1}{8} \cdot \sqrt{\frac{9}{2}} \cdot (7x^2 - 3) \cdot x; \quad \widetilde{l}_6(x) = \frac{1}{16} \cdot \sqrt{\frac{11}{2}} \cdot (21x^4 - 14x^2 + 1);$$

$$\widetilde{l}_7(x) = \frac{1}{16} \cdot \sqrt{\frac{13}{2}} \cdot (33x^4 - 30x^2 + 5) \cdot x;$$

$$\widetilde{l}_8(x) = \frac{1}{128} \cdot \sqrt{\frac{15}{2}} \cdot (429x^6 - 495x^4 + 135x^2 - 5);$$

$$\widetilde{l}_9(x) = \frac{1}{128} \cdot \sqrt{\frac{17}{2}} \cdot (715x^6 - 1001x^4 + 385x^2 - 35) \cdot x;$$

$$\widetilde{l}_{10}(x) = \frac{1}{256} \cdot \sqrt{\frac{19}{2}} \cdot (2431x^8 - 4004x^6 + 2002x^4 - 308x^2 + 7).$$

From our construction (4.5), it follows that $l'_k(x) = \dfrac{1}{\|L_{k-1}\|} \cdot L_{k-1}(x)$; thus,

$$\int_{-1}^{1} l'_i(x) \cdot l'_j(x)\,\mathrm{d}x = \frac{1}{\|L_{i-1}\|} \cdot \frac{1}{\|L_{j-1}\|} \cdot \int_{-1}^{1} L_{i-1}(x) \cdot L_{j-1}(x)\,dx. \tag{4.6}$$

The fact that the Legendre polynomials are orthogonal to each other means that

$$\int_{-1}^{1} L_{i-1}(x) \cdot L_{j-1}(x)\,dx = 0$$

for $i \neq j$ and thus, for $i \neq j$, we have $\int_{-1}^{1} l'_i(x) \cdot l'_j(x)\,\mathrm{d}x = 0$. For $i = j$, the above formula leads to

$$\int_{-1}^{1} (l'_i(x))^2\,\mathrm{d}x = \frac{1}{\|L_{i-1}\|^2} \cdot \int_{-1}^{1} L_{i-1}^2(x)\,dx. \tag{4.7}$$

By definition of $\|L_k\|$, this means that $\int_{-1}^{1} (l'_i(x))^2\,\mathrm{d}x = 1$. By combining the cases $i \neq j$ and $i = j$, we conclude that

$$\int_{-1}^{1} l'_i(x) \cdot l'_j(x)\,\mathrm{d}x = \delta_{ij}, \quad 2 \le i, j. \tag{4.8}$$

The functions $l_2, l_3, \ldots, l_p$ can be used as a basis in the space $V_{h,p} = P_0^p(\Omega)$ of all polynomials of order p that vanish at the endpoints of the interval $\Omega = (-1, 1)$. The desired solution $u_{h,p}$ belongs to this space $V_{h,p}$ and therefore, it can be represented as linear combination of the basis functions $l_2, \ldots, l_p$. If we denote the coefficient at l_{j+1} by y_j, we get the following representation:

$$u_{h,p}(x) = \sum_{j=1}^{p-1} y_j \cdot l_{j+1}(x). \tag{4.9}$$

We want to find the solution $u_{h,p}(x)$ which satisfies the condition (4.2) for all $w_{h,p} \in V_{h,p}$. Since the condition (4.2) is linear in $w_{h,p}$ and the functions $l_2, \ldots, l_p$ form a basis in the space $V_{h,p}$, it is sufficient to test the condition (4.2) only for these basis functions $w_{h,p}(x) = l_{i+1}(x)$. For each of these functions, the condition (4.2) takes the form

$$\int_{-1}^{1} u'_{h,p}(x) \cdot l'_{i+1}\,dx = \int_{-1}^{1} f_{h,p}(x) l_{i+1}(x)\,\mathrm{d}x. \tag{4.10}$$

Substituting (4.9) into this formula, we conclude that

$$\int_{-1}^{1} \left(\sum_{j=1}^{p-1} y_j \cdot l'_{j+1}(x) \right) \cdot l'_{i+1}(x)\,\mathrm{d}x =$$

$$\sum_{j=1}^{p-1} y_j \cdot \left(\int_{-1}^{1} l'_{j+1}(x) \cdot l'_{i+1}(x) \, \mathrm{d}x \right) = \int_{-1}^{1} f_{h,p}(x) \cdot l_{i+1}(x) \, \mathrm{d}x.$$

Due to (4.8), we conclude that the left-hand side of this equality is equal to y_i, hence

$$y_i = \int_{-1}^{1} f_{h,p}(x) \cdot l_{i+1}(x) \, \mathrm{d}x, \qquad 1 \leq i \leq p-1. \tag{4.11}$$

Substituting the values y_i from the equation (4.11) into the formula (4.9) for $u_{h,p}(x)$, we conclude that

$$u_{h,p}(x) = \sum_{i=1}^{p-1} \left(\int_{-1}^{1} f_{h,p}(z) \cdot l_{i+1}(z) \, \mathrm{d}z \right) \cdot l_{i+1}(x). \tag{4.12}$$

This formula can be rewritten as

$$u_{h,p}(x) = \int_{-1}^{1} f_{h,p}(z) \cdot \Phi_p(x,z) \, \mathrm{d}z, \tag{4.13}$$

where

$$\Phi_p(x,z) \stackrel{\text{def}}{=} \sum_{i=1}^{p-1} l_{i+1}(x) \cdot l_{i+1}(z). \tag{4.14}$$

Our objective is to show that the inverse operator $f_{h,p} \to u_{h,p}$ is nonnegativity preserving, i.e., that once $f_{h,p}(z)$ is non-negative for all $z \in (-1,1)$, the resulting solution $u_{h,p}(x)$ is nonnegative for all $x \in (-1,1)$. A possible way of proving this nonnegativity preservation property was provided by P. Šolín and T. Vejchodský in [201] as the following 2-step procedure:

1. Identify a subdomain Ω_p^+ of $(-1,1)$ where the function Φ_p is positive.

2. Find a quadrature rule of the order of accuracy $2p$ (exact for all polynomials of degree less or equal to $2p$) with positive weights and points lying in Ω_p^+.

The subdomains Ω_p^+ and the corresponding quadrature rules were constructed in [201]. So, to complete the proof of nonnegativity preservation, we must prove that each function $\Phi_p(x,y)$ is non-negative on the selected domain. For the quadratic case $p=2$ and for the cubic case $p=3$, this non-negativity was proven in [201]. The proof for higher values of p can be done by using interval computations.

4.6 Application of Interval Arithmetics to the Proof of Non-Negativity Preservation

4.6.1 Quartic case: first proof

Let us start with the quartic case $p = 4$. For this case, we want to prove that the function

$$\Phi_p(x, z) \stackrel{\text{def}}{=} \sum_{i=1}^{p-1} l_{i+1}(x) \cdot l_{i+1}(z)$$

is non-negative for all $x, z \in (-1, 1)$, where $l_i(x)$ are Lobatto shape functions

$$l_i(x) \stackrel{\text{def}}{=} (x^2 - 1) \cdot \widetilde{l}_i(x),$$

and

$$\widetilde{l}_2(x) = \frac{1}{2} \cdot \sqrt{\frac{3}{2}}; \;\; \widetilde{l}_3(x) = \frac{1}{2} \cdot \sqrt{\frac{5}{2}} \cdot x; \;\; \widetilde{l}_4(x) = \frac{1}{8} \cdot \sqrt{\frac{7}{2}} \cdot (5 \cdot x^2 - 1).$$

First of all, due to the above definition of $l_i(x)$, we can see that

$$\Phi_p(x, z) = (x^2 - 1) \cdot (z^2 - 1) \cdot \Psi_p(x, z),$$

where

$$\Psi_p(x, z) \stackrel{\text{def}}{=} \sum_{i=2}^{p} \widetilde{l}_i(x) \cdot \widetilde{l}_i(z).$$

When $x \in (-1, 1)$ and $z \in (-1, 1)$, we have $x^2 - 1 \leq 0$ and $z^2 - 1 \leq 0$, hence

$$(x^2 - 1) \cdot (z^2 - 1) \geq 0.$$

Therefore, to prove that $\Phi_4(x, z)$ is always non-negative, it is sufficient to prove that $\Psi_4(x, z) > 0$ for all $x, z \in [-1, 1]$.

By definition,

$$\Psi_4(x, z) = \widetilde{l}_2(x) \cdot \widetilde{l}_2(z) + \widetilde{l}_3(x) \cdot \widetilde{l}_3(z) + \widetilde{l}_4(x) \cdot \widetilde{l}_4(z) =$$

$$\frac{1}{4} \cdot \frac{3}{2} + \frac{1}{4} \cdot \frac{5}{2} \cdot x \cdot z + \frac{1}{64} \cdot \frac{7}{2} \cdot (5 \cdot x^2 - 1) \cdot (5 \cdot z^2 - 1) =$$

$$\frac{1}{128}\cdot(48+80\cdot x\cdot z+7\cdot(25\cdot x^2\cdot z^2-5\cdot x^2-5\cdot z^2+1))=$$
$$\frac{1}{128}\cdot(55+80\cdot x\cdot z-35\cdot x^2-35\cdot z^2+175\cdot x^2\cdot z^2).$$

So, the problem is to prove that

$$f(x,z)\stackrel{\text{def}}{=}55+80\cdot x\cdot z-35\cdot x^2-35\cdot z^2+175\cdot x^2\cdot z^2>0$$

for all $x,z\in[-1,1]$.

Before proving, let us further simplify the problem. Each value $x\in[-1,1]$ can be represented as $\varepsilon_x\cdot X$, where $X=|x|\in[0,1]$ and $\varepsilon_x=\text{sign}(x)=\pm1$. If we plug in the values $x=\varepsilon_x\cdot X$ and $z=\varepsilon_z\cdot Z$ into the formula for $f(x,z)$, then we get

$$f(x,z)=55+80\cdot\varepsilon_x\cdot\varepsilon_z\cdot X\cdot Z-35\cdot X^2-35\cdot Z^2+175\cdot X^2\cdot Z^2.$$

The only term that depends on the signs is the term coming from $80\cdot x\cdot z$, and it attains its smallest value when $\varepsilon_x\cdot\varepsilon_z=-1$. So, to prove that $f(x,z)>0$ for all $x,z\in[-1,1]$, it is sufficient to prove that

$$F(X,Z)\stackrel{\text{def}}{=}55-80\cdot X\cdot Z-35\cdot X^2-35\cdot Z^2+175\cdot X^2\cdot Z^2>0$$

for all $X,Z\in[0,1]$.

To prove this inequality, let us pick an integer n and subdivide the interval $[0,1]$ into n equal parts

$$\left[0,\frac{1}{n}\right],\left[\frac{1}{n},\frac{2}{n}\right],\ldots,\left[\frac{i}{n},\frac{i+1}{n}\right],\ldots,\left[\frac{n-1}{n},1\right].$$

Then, the square $[0,1]^2$ gets divided into n^2 small squares

$$[\underline{X},\overline{X}]\times[\underline{Z},\overline{Z}]=\left[\frac{i}{n},\frac{i+1}{n}\right]\times\left[\frac{j}{n},\frac{j+1}{n}\right].$$

The function $F(X,Z)$ is a sum of several monomial terms: $F(X,Z)=\sum\limits_{k=1}^{m}F_k(X,Z)$. If a monomial $F_k(X,Z)$ has a positive coefficient, then it is an increasing function of both X and Z, so it attains its minimum when $X=\underline{X}$ and $Z=\underline{Z}$. Hence, for all X and Z from the small box, we have $F_k(X,Z)\geq F_k(\underline{X},\underline{Z})$.

If a monomial $F_k(X, Z)$ has a negative coefficient, then it is an decreasing function of both X and Z, so it attains its minimum when $X = \overline{X}$ and $Z = \overline{Z}$. Hence, for all X and Z from the small box, we have $F_k(X, Z) \geq F_k(\overline{X}, \overline{Z})$.

Adding these inequalities, we conclude that for all $X, Z \in [\underline{X}, \overline{X}] \times [\underline{Z}, \overline{Z}]$, we have

$$F(X, Z) \geq 55 - 80 \cdot \overline{X} \cdot \overline{Z} - 35 \cdot \overline{X}^2 - 35 \cdot \overline{Z}^2 + 175 \cdot \underline{X}^2 \cdot \underline{Z}^2.$$

So, to prove that $F(X, Z) > 0$ for all $X, Z \in [0, 1]$, it is sufficient to prove that the right-hand side of the above inequality is positive for all n^2 small squares, i.e., that

$$55 - 80 \cdot \overline{X} \cdot \overline{Z} - 35 \cdot \overline{X}^2 - 35 \cdot \overline{Z}^2 + 175 \cdot \underline{X}^2 \cdot \underline{Z}^2 > 0.$$

Substituting the explicit expressions for $\underline{X}$, $\overline{X}$, $\underline{Z}$, and $\overline{Z}$, we conclude that we need to prove, for every i from 0 to $n - 1$ and for every j from 0 to $n - 1$, that

$$55 - 80 \cdot \frac{i+1}{n} \cdot \frac{j+1}{n} - 35 \cdot \frac{(i+1)^2}{n^2} - 35 \cdot \frac{(j+1)^2}{n^2} + 175 \cdot \left(\frac{i}{n}\right)^2 \cdot \left(\frac{j}{n}\right)^2 > 0.$$

To check this inequality for all n^2 combinations of i and j, we wrote a Java program that computes the values of all n^2 left-hand sides and returns the smallest of these values. For $n = 32$, the minimum is attained when $[\underline{X}, \overline{X}] = [0.3125, 0.34375]$ and $[\underline{Z}, \overline{Z}] = [0.96875, 1]$, and this minimum is equal to 4.402676. This is clearly a positive number.

We have selected n to be a power of 2 because in this case, the fractions are exactly represented inside the computer, so all additions and multiplications are exact. We have first tried $n = 2$, $n = 4$, $n = 8$, and $n = 16$, but for these n, the smallest bound is still negative, so $n = 32$ is the smallest value for which this proof works.

To be on a completely safe side, we repeated the computations by replacing floating-point numbers with integers. Specifically, we multiplied the above lower bound for $F(X, Z)$ by n^4. Then, this lower bound turns into an integer

$$55 \cdot n^4 - 80 \cdot (i+1) \cdot (j+1) \cdot n^2 - 35 \cdot (i+1)^2 \cdot n^2 - 35 \cdot (j+1)^2 \cdot n^2 + 175 \cdot i^2 \cdot j^2.$$

We ran another Java program to compute this expression for all i, j from 0 to $n - 1$ and to find the smallest of the resulting values. The smallest value is when $i = 10$, $j = 31$ (or when $i = 31$ and $j = 10$), and it is equal to $4616540 > 0$.

4.6.2 Second proof for the quartic case and the general case

The above proof was based on a very specific transformation applied to the quartic expression. To describe how similar techniques can handle a more general case, let us provide an alternative proof which is not using any such simplifying transformations at all and relies instead on the original formulas for the Lobatto shape functions.

In the quartic case $p = 4$, we deal with the function

$$\Phi_4(x, z) = \sum_{i=1}^{3} l_{i+1}(x) \cdot l_{i+1}(z).$$

Substituting the explicit expressions for the Lobatto shape functions into this formula, we conclude that

$$\Phi_4(x, z) = \sum_{i=1}^{3} (x^2 - 1) \cdot (z^2 - 1) \cdot \Psi_4(x, z),$$

where

$$\Psi_4(x, z) = \frac{3}{8} + \frac{5}{8} \cdot x \cdot z + \frac{7}{128} \cdot (5x^2 - 1) \cdot (5z^2 - 1). \tag{4.15}$$

We will prove that $\Phi_4(x, z) \geq 0$ for all $x, z \in (-1, 1)$ by using interval computations.

In interval computations, as we have mentioned earlier, one deals with intervals instead of numbers, and elementary arithmetic operations are extended from numbers to intervals in a natural way. For example, $[\underline{a}, \overline{a}] + [\underline{b}, \overline{b}] = [\underline{a} + \underline{b}, \overline{a} + \overline{b}]$, $[\underline{a}, \overline{a}] - [\underline{b}, \overline{b}] = [\underline{a} - \overline{b}, \overline{a} - \underline{b}]$, and so on. If we replace every operation with numbers by the corresponding operation of interval arithmetic, we get an enclosure for the range of the analyzed function on given intervals [94].

Let us use this technique to prove the nonnegativity of the function (4.15) in the square $[-1, 1]^2$: Substituting a pair of intervals $X = [\underline{x}, \overline{x}]$ and $Z = [\underline{z}, \overline{z}]$ into (4.15), we obtain an enclosure

$$[\underline{\Psi}_4, \overline{\Psi}_4] \supseteq \Psi_4(X, Z) = \{\Psi_4(x, z);\ x \in X, z \in Z\}.$$

Moreover, since the function (4.15) is polynomial and it only contains rational coefficients, its evaluation for rational intervals can be done using exact integer arithmetic.

Consider the intervals $X_1 = Z_1 = [-1, 1]$. This box is shown in Figure 4.1; the yellow color means that we do not know the enclosure $[\underline{\Psi}_4, \overline{\Psi}_4]$ for $\Psi_4(X_1, Z_1)$ yet.

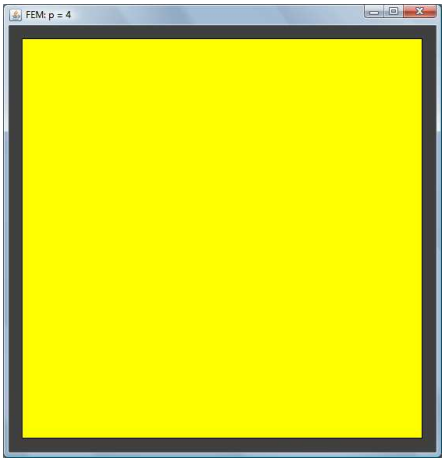


Figure 4.1: Step 1: Ψ_4, Box $X_1 = Z_1 = [-1, 1]$.

Step 1: The program computes:

$$[\underline{\Psi}_4, \overline{\Psi}_4] = [-25/16, 95/32] \supseteq \Psi_4(X_1, Z_1).$$

If the left endpoint $\underline{\Psi}_4$ of the enclosure interval $[\underline{\Psi}_4, \overline{\Psi}_4]$ was nonnegative, then the proof would be finished. Since this is not the case, we refine the grid by halving both the intervals X_1 and Z_1. We obtain four subdomains $[-1, 0] \times [-1, 0]$, $[-1, 0] \times [0, 1]$, $[0, 1] \times [-1, 0]$, and $[0, 1] \times [0, 1]$. This new grid is shown in Figure 4.2

Step 2: The program computes the enclosures for these subdomains:

- for $[-1, 0] \times [-1, 0]$, we get $[\underline{\Psi}_4, \overline{\Psi}_4] = [5/32, 15/8] \supseteq \Psi_4([-1, 0], [-1, 0])$;
- for $[-1, 0] \times [0, 1]$, we get $[\underline{\Psi}_4, \overline{\Psi}_4] = [-15/32, 5/4] \supseteq \Psi_4([-1, 0], [0, 1])$;
- for $[0, 1] \times [-1, 0]$, we get $[\underline{\Psi}_4, \overline{\Psi}_4] = [-15/32, 5/4] \supseteq \Psi_4([0, 1], [-1, 0])$;
- for $[0, 1] \times [0, 1]$, we get $[\underline{\Psi}_4, \overline{\Psi}_4] = [5/32, 15/8] \supseteq \Psi_4([0, 1], [0, 1])$.

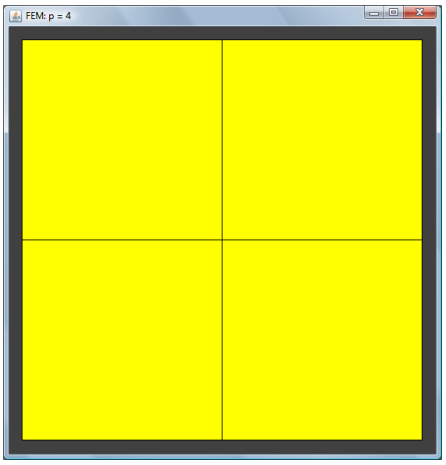


Figure 4.2: Step 2: Ψ_4, Boxes $[-1,0]\times[-1,0]$, $[-1,0]\times[0,1]$, $[0,1]\times[-1,0]$, and $[0,1]\times[0,1]$.

This proves that the function Ψ_4 (and hence also Φ_4) is nonnegative in the subdomains $[-1,0]\times[-1,0]$ and $[0,1]\times[0,1]$. This is shown in Figure 4.3, where these boxes are shown in blue, which means the enclosure $[\underline{\Psi}_4, \overline{\Psi}_4]$ is non-negative for these boxes. As for the remaining subdomains $[-1,0]\times[0,1]$ and $[0,1]\times[-1,0]$, we divide each of them into four equal subdomains, compute the enclosure for each new subdomain, etc.

After six iterations of this procedure, we get a partition of $[-1,1]^2$ for which the left endpoints of the enclosures are nonnegative, as shown in Figure 4.4. So we have proved that Ψ_4 (and hence also Φ_4) is nonnegative in $[-1,1]^2$.

A similar proof works for $p = 5,\ldots,10$. The Java programs and output files with details on the computations for $p = 4,5,\ldots,10$ are available at
http://www.math.utep.edu/Faculty/solin/intcomp.

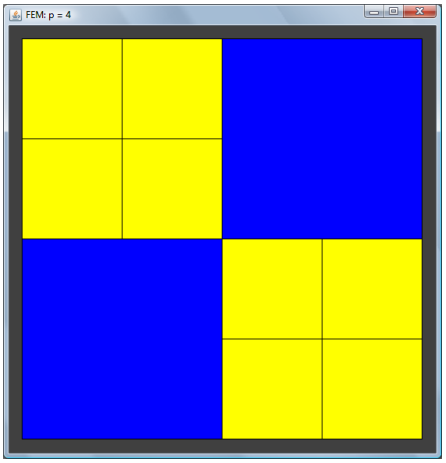


Figure 4.3: Step 3: Ψ_4 Boxes $[-1,0] \times [-1,0]$ and $[0,1] \times [0,1]$ are non-negative, boxes $[-1,0] \times [0,1]$, $[0,1] \times [-1,0]$ are partitioned again.

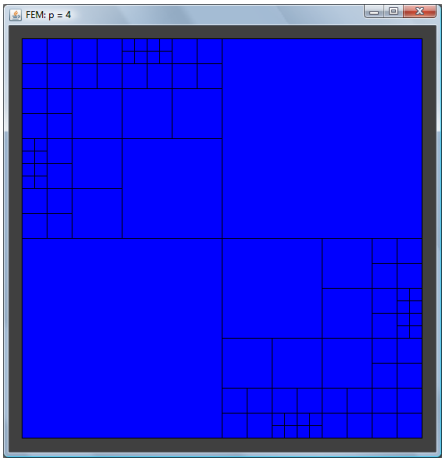


Figure 4.4: Step 2: Ψ_4 is non-negative on Box $X_1 = Z_1 = [-1,1]$.

Chapter 5

Conclusions and Future Work

General formulation of the problem. The more data we need to process, the more computing power we need and thus, the more important it is to use (and design) faster algorithms for data processing. Even processing 1-D data often is very time-consuming, but processing 2-D data (e.g., images) and 3-D data is where we really encounter the limits of the current computer hardware abilities. It is therefore desirable to speed up processing of 2-D and 3-D data.

One possible way to speed up data processing algorithms is to take into account the knowledge that the experts have about the domain, the knowledge that was not taken into account in the existing algorithms. Taking this knowledge into account is also important because the existing data processing algorithms sometimes produce results which contradict the expert knowledge. For example, a mathematical solution to a seismic inverse problem may lead to un-physically large values of density inside the Earth.

One of the common forms of expert knowledge is the knowledge of the bounds (i.e., intervals) on the actual values of the physical quantities. It is therefore desirable to use this interval-related expert knowledge in processing 2-D and 3-D data.

What we do. In this dissertation, we have shown that interval-related expert knowledge can be used in all types of 2-D and 3-D data processing problems.

Simplest case of data processing and the related referencing problem. In general, an input to the corresponding data processing algorithm is a 2-D or 3-D data set – which consists of the values of certain quantities $q(\vec{x})$ at different spatial locations $\vec{x}$.

The simplest case is when these values $q(\vec{x})$ are directly measurable – a typical example of such a situation is a satellite image. In this simplest case, usually, the main remaining problem is that the spatial locations $\vec{x}$ are not known exactly. Thus, to combine several data sets describing the same object, we must "match" (*reference*) the corresponding images. There exist efficient algorithms for referencing 2-D images. In this dissertation, we use expert knowledge about smoothness or abruptness of the corresponding images to extend these 2-D algorithms to the more complex 3-D situation. Specifically, we consider situations in which the images-to-be-matched differ from each other only by shifts and rotations.

Inverse problems: a general case of data processing. In practice, often, we cannot directly measure the values $q(\vec{x})$, we need to find them by first measuring the values of some related quantity (or possibly several related quantities) and then by solving the corresponding equations – i.e., by solving the corresponding *inverse problem*.

In this dissertation, we show, on the example of the seismic inverse problem in geophysics, how interval-related expert knowledge can help in solving such a problem. Specifically, we get the solution which is in better accordance with the interval-related expert knowledge, and we get it faster than by using the existing ad hoc algorithms.

How close is the solution of the inverse problem to the actual image? Once we have found a solution to the inverse problem, we need to estimate how close this solution is to the actual image. In this dissertation, on the example of a simple 1-D inverse problem, we show how interval techniques can help in this estimation.

General conclusion. In all the problems related to processing 2-D and 3-D data, the use of interval-related expert knowledge can be very beneficial and productive.

Future work. In this dissertation, we have shown, on simple practically useful examples, that interval-related expert knowledge can be used in processing 2-D and 3-D data. It is desirable to extend these results to more general situations.

For *referencing*, it is desirable to extend our algorithms:

- to generic images, i.e., images which may contain both smooth and abrupt parts, and
- to generic situations, i.e., situations like robotic vision and aerial images, when in addition to shift, rotation, and scaling, we also have distortions caused by different perspectives.

For *solving inverse problems*, it is desirable to extend our algorithm to generic inverse problems and generic interval-related constraints.

For *estimating how close* is the solution of the inverse problem to the actual image, it is desirable to extend our methods to more complex problems.

References

[1] M. Abrams, S. Hook, and B. Ramachandran. *Aster User Handbook*. NASA Jet Propulsion Laboratory, Pasadena, CA, 2002.

[2] J. Aczel. *Lectures on Functional Equations and Their Applications*. Academic Press, New York, 1966.

[3] J.E. Adidhela. Using FFT-based data processing techniques to characterize asphaltic concrete mixtures. Master's thesis, The University of Texas at El Paso, Department of Computer Science, 2004.

[4] M. Ainsworth and B. Senior. Aspects of an *hp*-adaptive finite element method: Adaptive strategy, conforming approximation and efficient solvers. Technical Report 1997/2, University of Leicester, Department of Mathematics and Computer Science, England, U.K., 1997.

[5] U.O. Akpan, T.S. Koko, I.R. Orisamolu, and B.K. Gallant. Practical fuzzy finite element analysis of structures. *Finite Elem. Anal. Des.*, 38:93–111, 2001.

[6] S.H. Alavi and C.L. Monismith. Time and temperature dependent properties of asphalt concrete mixes tested as hollow cylinders and subjected to dynamic axial and shear loads. *Asphalt Paving Technology*, 63:152–181, 1994.

[7] R. Aldouri, G.R. Keller, A. Gates, J. Rasillo, L. Salayandia, V. Kreinovich, J. Seeley, P. Taylor, and S. Holloway. Geon: Geophysical data add the 3rd dimension in geospatial studies. In *Proceedings of the ESRI International User Conference 2004*, San Diego, CA, August 9–13 2004. Paper 1898.

[8] C. Anderson and M. D. Dahleh. Rapid computation of the discrete fourier transform. *SIAM J. Sci. Computing*, 17:913–919, 1996.

[9] R. Araiza, H. Xie, S.A. Starks, and V. Kreinovich. Automatic referencing of multi-spectral images. In *Proceedings of the IEEE Southwest Symposium on Image Analysis and Interpretation*, pages 21–25, Santa Fe, NM, April 7–9 2002.

[10] R. Araiza. Optimal FFT-based algorithms for referencing multi-spectral images. Master's thesis, The University of Texas at El Paso, Department of Computer Science, 2003.

[11] R. Araiza, M.G. Averill, G.R. Keller, S.A. Starks, and C. Bajaj. 3-d image registration using fast fourier transform, with potential applications to geoinformatics and bioinformatics. In *Proceedings of the International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems IPMU'06*, pages 817–824, Paris, France, July 2–7 2006.

[12] M.G. Averill, K.C. Miller, G.R. Keller, V. Kreinovich, J. Beck, R. Araiza, R. Torres, and S.A. Starks. Monte-carlo-type techniques for processing interval uncertainty, and their geophysical and engineering applications. Technical Report UTEP-CS-04-40, The University of Texas at El Paso, Department of Computer Science, 2004.

[13] M.G. Averill, K.C. Miller, G.R. Keller, V. Kreinovich, R. Araiza, and S.A. Starks. Using expert knowledge in solving the seismic inverse problem. In *Proceedings of the 24nd International Conference of the North American Fuzzy Information Processing Society NAFIPS'2005*, pages 310–314, Ann Arbor, MI, June 22–25 2005.

[14] M.G. Averill, K.C. Miller, G.R. Keller, V. Kreinovich, R. Araiza, and S.A. Starks. Using expert knowledge in solving the seismic inverse problem. *International Journal of Approximate Reasoning.* (to appear).

[15] I. Babuška and W. Gui. The h, p, and hp-versions of the finite element method in one dimension. parts i–iii. *Numer. Math.*, 49:577–683, 1986.

[16] I. Babuška, M. Griebel, and J. Pitkäranta. The problem of selecting the shape functions for p-type elements. *Int. J. Numer. Meth. Engrg.*, 28:1891–1908, 1999.

[17] I. Babuška and T.J. Oden. Verification and validation in computational engineering and science: Basic concepts. *Computer Methods in Applied Mechanics and Engineering*, 193:4057–4066, 2004.

[18] C. Bajaj, J. Castrillon-Candas, V. Siddavanahalli, and Z. Xu. Compressed representations of macromolecular structures and properties structure. *Structure*, 13(3):463–471, 2005.

[19] C. Bajaj and V. Siddavanahalli. Fast error-bounded surfaces and derivatives computation for volumetric particle data. Technical Report TR-06-06, The University of Texas at Austin, Department of Computer Sciences, January 2006. http://www.cs.utexas.edu/ftp/pub/techreports/tr06-06.pdf.

[20] G. Bardossy and J. Fodor. *Evaluation of Uncertainties and Risks in Geology*. Springer Verlag, Berlin, 2004.

[21] B. Beck, V. Kreinovich, and B. Wu. Interval-valued and fuzzy-valued random variables: From computing sample variances to computing sample covariances. In M. Lopez, M. A. Gil, P. Grzegorzewski, O. Hrynewicz, and J. Lawry, editors, *Soft Methodology and Random Information Systems*, pages 85–92. Springer-Verlag, Berlin Heidelberg New York Tokyo, 2004.

[22] J. Beck, V. Kreinovich, and B. Wu. Interval-valued and fuzzy-valued random variables: From computing sample variances to computing sample covariances. In M. Lopez, M. A. Gil, P. Grzegorzewski, O. Hrynewicz, and J. Lawry, editors, *Soft Methodology and Random Information Systems*, pages 85–92. Springer-Verlag, Berlin-Heidelberg, 2004.

[23] T. Belytschko, N. Moës, S. Usui, and C. Parimi. Arbitrary discontinuities in finite elements. *International Journal of Numerical Methods in Engineering*, 50(4):993–1013, 2001.

[24] Y. Ben-Haim and I. Elishakoff. *Convex Models of Uncertainty in Applied Mechanics*. Elsevier Science, Amsterdam, 1990.

[25] D. Berleant. Automatically verified arithmetic with both intervals and probability density functions. *Interval Computations*, (2):48–70, 1993.

[26] D. Berleant. Automatically verified arithmetic on probability distributions and intervals. In R.B. Kearfott and V. Kreinovich, editors, *Applications of Interval Computations*. Kluwer, Dordrecht, 1996.

[27] D. Berleant and C. Goodman-Strauss. Bounding the results of arithmetic operations on random variables of unknown dependency using intervals. *Reliable Computing*, 4(2):147–165, 1998.

[28] D. Berleant, L. Xie, and J. Zhang. Statool: A tool for distribution envelope determination (denv), an interval-based algorithm for arithmetic on random variables. *Reliable Computing*, 9(2):91–108, 2003.

[29] G. Beylkin. On the fast fourier transform of functions with singularities. *Appl. Comput. Harmon. Anal.*, 2:363–381, 1995.

[30] D. Bizyaev, A.A. Zasov, and S. Kajsin. Combined color indexes and photometric structures of galaxies NGC 834 and NGC 1134. In F. Favata, A.A. Kass, and A. Wilson, editors, *Proc. of 33 ESLAB Symposium on Star Formation from the Small to the Large Scale*, European Space Agency, 2000.

[31] R.N. Bracewell. *The Fourier Transform and its Applications*. McGraw-Hill, New York, 1965.

[32] L.G. Brown. A survey of image registration techniques. *ACM Computing Surveys*, 24(4):325–376, 1992.

[33] R. Burgmann, P.A. Rosen, and E.J. Fielding. Synthetic aperture radar interferometry to measure earth's surface topography and its deformation. *Annu. Rev. Earth Planet. Sci.*, 28:169–209, 2000.

[34] S.D. Cabrera, K. Iyer, G. Xiang, and V. Kreinovich. On inverse halftoning: Computational complexity and interval computations. In *Proceedings of the 39th Conference on Information Sciences and Systems CISS'2005*, John Hopkins University, March 16–18 2005. Paper 164.

[35] J. Castrillon-Candas, C. Bajaj, and V. Siddavanahalli. An adaptive irregularly spaced fourier method for protein-protein docking. Technical Report TR-05-34, The University of Texas at Austin, Department of Computer Sciences, July 2005. http://www.cs.utexas.edu/ftp/pub/ techreports/tr05-34.pdf.

[36] Q. Chen, M. Defrise, and F. Deconnick. Symmetric phase-only matched filtering of fourier-mellin transform for image registration and recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16:1156–1168, 1994.

[37] S.H. Chen, H.D. Lian, and X.W. Yang. Interval static displacement analysis for structures with interval parameters. *International Journal of Numerical Methods in Engineering*, 53:393–407, 2002.

[38] H. Chen and T. Belytschko. An enriched finite element method for elastodynamic crack propagation. *International Journal of Numerical Methods in Engineering*, 58(12):1873–1905, 2003.

[39] J. Chessa, P. Smolinski, and T. Belytschko. The extended finite element method (X-FEM) for solidification problems. *International Journal of Numerical Methods in Engineering*, 53:1959–1977, 2002.

[40] J. Chessa, H. Wang, and T. Belytschko. On the construction of blending elements for local partition of unity enriched finite elements. *International Journal of Numerical Methods in Engineering*, 57(7):1015–1038, 2003.

[41] J. Chessa and T. Belytschko. An extended finite element method for two-phase fluids. *J. Appl. Mech.*, 70(1):10–17, 2003.

[42] J. Chessa and T. Belytschko. A local space-time discontinuous finite element method. *Computer Methods in Applied Mechanics and Engineering*. (to appear).

[43] P.G. Ciarlet. Discrete maximum principle for finite difference operators. *Aequationes Math.*, 4:338–352, 1970.

[44] P.G. Ciarlet and P.A. Raviart. Maximum principle and uniform convergence for the finite element method. *Computer Methods in Applied Mechanics and Engineering*, 2:17–31, 1973.

[45] A.V. Cideciyan. Registration in occular fundus images. *IEEE Engineering in Medicine and Biology*, 14:52–58, 1995.

[46] G. Corliss, C. Foley, and R.B. Kearfott. Formulation for reliable analysis of structural frames. In R.L. Muhanna and R.L. Mullen, editors, *Proc. NSF Workshop on Reliable Engineering Computing*, Savannah, GA, 2004. http://www.gtsav.gatech.edu/rec/recworkshop/index.html.

[47] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, Cambridge, MA, 2001.

[48] R.E. Crippen and R.G. Blom. Measurement of subresolution terrain displacements using spot panchromatic imagery. In *Proceedings of the 1991 International Geoscience and Remote Sensing Symposium IGARSS'91 "Remote Sensing: Global Monitoring for Earth Management"*, volume 3, pages 1667–1670, June 3–6 1991.

[49] R.A. Crowther. Procedures for three-dimensional reconstruction of spherical viruses by fourier synthesis from electron micrographs. *Phil. Trans. Roy. Soc. London*, 261:221–270, 1971.

[50] E. De Castro and C. Morandi. Registration of translated and rotated images using finite fourier transforms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(5):700–703, 1986.

[51] R. Demicco and G. Klir, editors. *Fuzzy Logic in Geology*. Academic Press, 2003.

[52] L. Demkowicz, W. Rachowicz, and P. Devloo. A fully automatic *hp*-adaptivity. Technical Report TICAM 01-28, The University of Texas at Austin, 2001.

[53] A.P. Dempster. Upper and lower probabilities induced by a multi-valued mapping. *Ann. Mat. Stat.*, 38:325–339, 1967.

[54] O. Dessombz, F. Thouverez, J.-P. Laîné, and L. Jézéquel. Analysis of mechanical systems using interval computations applied to finite elements methods. *Sound. Vib.*, 238(5):949–968, 2001.

[55] S. Dominguez, J.P. Avouac, and R. Michel. Horizontal coseismic deformation of the 1999 chi-chi earthquake measured from spot satellite images: Implications for the seismic cycle along the western foothills of central taiwan. *Journal of Geophysical Research*, 108(B2):2083, 2003.

[56] D.I. Doser, K.D. Crain, M.R. Baker, V. Kreinovich, and M.C. Gerstenberger. Estimating uncertainties for geophysical tomography. *Reliable Computing*, 4(3):241–268, 1998.

[57] A. Dravskikh, A.M. Finkelstein, and V. Kreinovich. Astrometric and geodetic applications of VLBI 'arc method'. In *Modern Astrometry, Proceedings of the IAU Colloquium No. 48*, pages 143–153, Vienna, 1978.

[58] A.F. Dravskikh, O.M. Kosheleva, V. Kreinovich, and A.M. Finkelstein. The method of arcs and differential astrometry. *Soviet Astronomy Letters*, 5(3):160–162, 1979.

[59] A. Dutt and V. Rokhlin. Fast fourier transform for nonequispaced data. *SIAM J. Sci. Computing*, 14:1368–1393, 1993.

[60] A. Dutt and V. Rokhlin. Fast fourier transform for nonequispaced data. ii. *Appl. Comput. Harmon. Anal.*, 2:85–100, 1995.

[61] B. Elbel. Fast fourier transform for non-equispaced data. In C.K. Chui and L.L. Schumaker, editors, *Approximation Theory*. Vanderbilt University Press, 1998.

[62] I. Elishakoff and Y. Ren. The bird's eye view on finite element method for stochastic structures. *Computer Methods in Applied Mechanics and Engineering*, 168:51–61, 1999.

[63] K.L. Feigl, F. Sarti, H. Vadon, S. McClusky, S. Ergintav, P. Durand, R. Burgmann, A. Rigo, D. Massonnet, and R. Reilinger. Estimating slip distribution for the izmit mainshock from coeseismic GPS, ERS-1, RADARSAT, and SPOT measurements. *Bulletin of the Seismological Society of America*, 92(1):138–160, 2002.

[64] S. Ferson and L.R. Ginzburg. Different methods are needed to propagate ignorance and variability. *Reliab. Engng. Syst. Saf.*, 54:133–144, 1996.

[65] S. Ferson, D. Myers, and D. Berleant. Distribution-free risk analysis: I. range, mean, and variance. Technical report, Applied Biomathematics, 2001.

[66] S. Ferson. *RAMAS Risk Calc 4.0: Risk Assessment with Uncertain Numbers*. CRC Press, Boca Raton, FL, 2002.

[67] S. Ferson, L. Ginzburg, V. Kreinovich, and M. Aviles. Exact bounds on sample variance of interval data. In *Extended Abstracts of the 2002 SIAM Workshop on Validated Computing*, pages 67–69, Toronto, Canada, 2002.

[68] S. Ferson, L. Ginzburg, V. Kreinovich, and M. Aviles. Exact bounds on finite populations of interval data. Technical Report UTEP-CS-02-13d, The University of Texas at El Paso, Department of Computer Science, 2002. http://www.cs.utep.edu/vladik/2002/tr02-13d.pdf.

[69] S. Ferson, L. Ginzburg, V. Kreinovich, L. Longpré, and M. Aviles. Computing variance for interval data is np-hard. *ACM SIGACT News*, 33(2):108–118, 2002.

[70] S. Ferson, V. Kreinovich, L. Ginzburg, D.S. Myers, and K. Sentz. Constructing probability boxes and Dempster-Shafer structures. Technical Report SAND2002-4015, Sandia National Laboratories, 2003.

[71] S. Ferson, L. Ginzburg, V. Kreinovich, L. Longpré, and M. Aviles. Exact bounds on finite populations of interval data. *Reliable Computing*, 11(3):207–233, 2005.

[72] P.C. Fishburn. *Utility Theory for Decision Making*. John Wiley & Sons Inc., New York, 1969.

[73] P.C. Fishburn. *Nonlinear Preference and Utility Theory*. John Hopkins Press, Baltimore, MD, 1988.

[74] H. Foroosh, J.B. Zerubia, , and M. Berthod. Extension of phase correlation to subpixel registration. *IEEE Transactions on Image Processing*, 11:188–200, 2002.

[75] S. Ganzerli and C.P. Pantelides. Load and resistance convex models for optimum design. *Struct. Optim.*, 17:259–268, 1999.

[76] F. Garcia, R. Araiza, and B. Rzycki. Towards optimal mosaicking of multi-spectral images. In *Proc. 2nd Nat'l NASA Student Conference*, Nashville, TN, April 7–10 2000.

[77] M.E. Garey and D.S. Johnson. *Computers and intractability: a guide to the theory of NP-completeness*. Freeman, San Francisco, CA, 1979.

[78] D.J.H. Garling. A book review. *American Mathematical Monthly*, 112(6):575–579, 2005.

[79] S. Gibson. An optimal fft-based algorithm for mosaicking images. Master's thesis, The University of Texas at El Paso, Department of Computer Science, 1999.

[80] S. Gibson, V. Kreinovich, L. Longpré, B. Penn, and S.A. Starks. Intelligent mining in image databases, with applications to satellite imaging and to web search. In A. Kandel, H. Bunke, and M. Last, editors, *Data Mining and Computational Intelligence*, pages 309–336. Springer-Verlag, Berlin, 2001.

[81] L. Granvilliers, V. Kreinovich, and N. Müller. Novel approaches to numerical software with result verification. In R. Alt, A. Frommer, R. B. Kearfott, and W. Luther, editors, *Numerical Software with Result Verification*, volume 2991, pages 274–305, International Dagstuhl Seminar, Dagstuhl Castle, Germany, January 19–24, 2003, 2004. Revised Papers, Springer Lectures Notes in Computer Science.

[82] L.K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the 28th Annual ACM Symposium on the Theory of Computing STOC'96*, pages 212–219, 1996.

[83] L.K. Grover. From schrödinger's equation to quantum search algorithm. *American Journal of Physics*, 69(7):769–777, 2001.

[84] A. Haldar and S. Mahadevan. *Reliability Assessment Using Stochastic Finite Element Analysis*. John Wiley & Sons, New York, 2000.

[85] E. Hansen. *Global Optimization Using Interval Analysis*. Marcel Dekker, Inc., New York, 1992.

[86] G.H. Hardy, J.E. Littlewood, and G. Pólya. *Inequalities*. Cambridge University Press, 1988.

[87] D.R. Hiltunen and R.Roque. A mechanics-based prediction model for thermal cracking of asphaltic concrete pavements. *Asphalt Paving Technology*, 63:81–117, 1994.

[88] D.R. Hiltunen and R. Roque. The use of time-temperature superposition to fundamentally characterize asphaltic concrete mixtures at low temperatures. In G.A. Huber and D.S. Decker, editors, *Engineering Properties of Asphalt Mixtures and the Relationship to their Performance*, volume STP-1265. American Society for Testing and Materials, Philadelphia, PA, 1995.

[89] J.A. Hole. Nonlinear high-resolution three-dimensional seismic travel time tomography. *J. Geophysical Research*, 97(B5):6553–6562, 1992.

[90] W. Höhn and H.D. Mittelmann. Some remarks on the discrete maximum principle for finite elements of higher-order. *Computing*, 27:145–154, 1981.

[91] Y.H. Huang. *Pavement Analysis and Design*. Prentice Hall, Englewood Cliffs, NJ, 1993.

[92] P.J. Huber. *Robust statistics*. Wiley, New York, 2004.

[93] J. Jájá. *An Introduction to Parallel Algorithms*. Addison-Wesley, Reading, MA, 1992.

[94] L. Jaulin, M. Kieffer, O. Didrit, and E. Walter. *Applied Interval Analysis*. Springer Verlag, London, 2001.

[95] A. Johnson, A. Ansar, and L. Matthies. Mapped landmark algorithm for precision landing. *NASA Software Tech Briefs*, page 27, September 2007.

[96] J. Karátson and S. Korotov. Discrete maximum principles for finite element solutions of nonlinear elliptic problems with mixed boundary conditions. *Numer. Math.*, 99:669–698, 2005.

[97] G.E. Karniadakis and S.J. Sherwin. *Spectral/hp Element Methods for CFD*. Oxford University Press, Oxford, 1999.

[98] R.B. Kearfott and V. Kreinovich. Beyond convex? global optimization is feasible only for convex objective functions: A theorem. *Journal of Global Optimization*, 33(4):617–624, 2005.

[99] G.R. Keller, T.G. Hildenbrand, R. Kucks, M. Webring, A. Briesacher, K. Rujawitz, A.M. Hittleman, D.R. Roman, D. Winester, R. Aldouri, J. Seeley, J. Rasillo, R. Torres, W.J. Hinze, A. Gates, V. Kreinovich, and L. Salayandia. A community effort to construct a gravity database for the united states and an associated web portal. In A.K. Sinha, editor, *Geoinformatics: Data to Knowledge*, pages 21–34. Geological Society of America Publ., Boulder, CO, 2006.

[100] G.R. Keller, S.A. Starks, A. Velasco, M. Averill, R. Araiza, G. Xiang, and V. Kreinovich. Towards combining probabilistic, interval, fuzzy uncertainty, and constraints: on the example of inverse problem in geophysics. In *Proceedings of the Second International Conference on Fuzzy Sets and Soft Computing in Economics and Finance FSSCEF'2006*, pages 47–54, St. Petersburg, Russia, June 28 - July 1 2006.

[101] D.G. Kendall. Foundations of a theory of random sets. In E. Harding and D. Kendall, editors, *Stochastic Geometry*, pages 322–376. New York, 1974.

[102] G. King, Y. Klinger, D. Bowman, and P. Tapponnier. Slip-partitioned surface breaks for the $m_w = 7.8$ 2001 kokoxili earthquake, china. *Bulletin of the Seismological Society of America*, 95:731–738, 2005.

[103] G. Klir and B. Yuan. *Fuzzy sets and fuzzy logic: theory and applications*. Prentice Hall, Upper Saddle River, NJ, 1995.

[104] O. Kosheleva, V. Kreinovich, and H.T. Nguyen. On the optimal choice of quality metric in image compression. In *Proceedings of the IEEE Southwest Symposium on Image Analysis and Interpretation*, pages 116–120, Santa Fe, NM, April 7–9 2002.

[105] V. Kreinovich, A. Bernat, O. Kosheleva, and A. Finkelstein. Interval estimates for closure phase and closure amplitude imaging in radio astronomy. *Interval Computations*, 2(4):51–71, 1992.

[106] V. Kreinovich, J. Pierluissi, and M. Koshelev. A new method of measuring strong currents by their magnetic fields. *Computers & Electrical Engineering*, 23(2):121–128, 1997.

[107] V. Kreinovich, A. Lakeyev, J. Rohn, and P. Kahl. *Computational complexity and feasibility of data processing and interval computations*. Kluwer, Dordrecht, 1997.

[108] V. Kreinovich, P. Patangay, L. Longpré, S.A. Starks, C. Campos, S. Ferson, and L. Ginzburg. Outlier detection under interval and fuzzy uncertainty: Algorithmic solvability and computational complexity. In *Proceedings of the 22nd International Conference of the North American Fuzzy Information Processing Society NAFIPS'2003*, pages 401–406, Chicago, IL, July 24–26 2003.

[109] V. Kreinovich and L. Longpré. Computational complexity and feasibility of data processing and interval computations, with extension to cases when we have partial information about probabilities. In V. Brattka, M. Schroeder, K. Weihrauch, and N. Zhong, editors, *Proc. Conf. on Computability and Complexity in Analysis CCA'2003*, pages 19–54, Cincinnati, OH, August 28–30 2003.

[110] V. Kreinovich and L. Longpré. Fast quantum algorithms for handling probabilistic and interval uncertainty. *Mathematical Logic Quarterly*, 50(4/5):507–518, 2004.

[111] V. Kreinovich, L. Longpré, S. Ferson, and L. Ginzburg. Computing higher central moments for interval data. Technical Report UTEP-CS-03-14b, The University of Texas at El Paso, Department of Computer Science, 2004. http://www.cs.utep.edu/vladik/2003/tr03-14b.pdf.

[112] V. Kreinovich, L. Longpré, P. Patangay, S. Ferson, and L. Ginzburg. Outlier detection under interval uncertainty: Algorithmic solvability and computational complexity. In I. Lirkov, S. Margenov, J. Wasniewski, and P. Yalamov, editors, *Large-Scale Scientific Computing, Proceedings of the 4-th International Conference LSSC'2003, June 4–8, 2003*, volume 2907, pages 238–245, Sozopol, Bulgaria, 2004. Springer Lecture Notes in Computer Science.

[113] V. Kreinovich. Probabilities, intervals, what next? optimization problems related to extension of interval computations to situations with partial information about probabilities. *Journal of Global Optimization*, 29(3):265–280, 2004.

[114] V. Kreinovich, J. Beck, C. Ferregut, A. Sanchez, G.R. Keller, M. Averill, and S.A. Starks. Monte-carlo-type techniques for processing interval uncertainty, and their engineering applications. In *Proceedings of the Workshop on Reliable Engineering Computing*, pages 139–160, Savannah, GA, September 15–17 2004.

[115] V. Kreinovich and S. Ferson. A new cauchy-based black-box technique for uncertainty in risk analysis. *Reliability Engineering and Systems Safety*, 85(1–3):267–279, 2004.

[116] V. Kreinovich, L. Longpré, P. Patangay, S. Ferson, and L. Ginzburg. Outlier detection under interval uncertainty: Algorithmic solvability and computational complexity. *Reliable Computing*, 11(1):59–76, 2005.

[117] V. Kreinovich, H.T. Nguyen, and B. Wu. On-line algorithms for computing mean and variance of interval data, and their use in intelligent systems. *Information Sciences*. (in press).

[118] V. Kreinovich, G. Xiang, S.A. Starks, L. Longpre, M. Ceberio, R. Araiza, J. Beck, R. Kandathi, A. Nayak, R. Torres, and J. Hajagos. Towards combining probabilistic and interval uncertainty in engineering calculations: algorithms for computing statistics under interval uncertainty, and their computational complexity. *Reliable Computing*. (to appear).

[119] S. Korotov, M. Křížek, and P. Neittaanmäki. Weakened acute type condition for tetrahedral triangulations and the discrete maximum principle. *Math. Comp.*, 70:107–119, 2000.

[120] U. Koyluoglu, S. Cakmak, N. Ahmet, and R. K. Soren. Interval algebra to deal with pattern loading and structural uncertainty. *J. Engrg. Mech.*, 121(11):1149–1157, 1995.

[121] M. Křížek and L. Liu. On the maximum and comparison principles for a steady-state nonlinear heat conduction problem. *ZAMM Z. Angew. Math. Mech.*, 83:559–563, 2003.

[122] C.D. Kuglin and D.C. Hines. The phase correlation image alignment method. In *Proc. IEEE International Conference on Cebrnetics and Society.*

[123] V.P. Kuznetsov. *Interval Statistical Models.* Radio i Svyaz, Moscow, 1991. (in Russian).

[124] H.R. Lang. *Algorithm Theoretical Basis Document for ASTER Digital Elevation Models (Standard Product AST14).* NASA Jet Propulsion Laboratory, 3.0 edition, 1999.

[125] T.M. Lehmann. A two stage algorithm for model-based registration of medical images. In *Proceedings of the International Conference on Pattern Recognition ICPR'98*, pages 344–352, Brisbane, Australia, 1998.

[126] A. Lin, B. Fu, J. Guo, Z. Qingli, D. Guangming, W. He, and Y. Zhao. Coseismic strike-slip and rupture length produced by the 2001 $m_s = 8.1$ central kunlun earthquake. *Science*, 296(5575):2015–2017, 2002.

[127] W.K. Liu, T. Belytschko, and A. Mani. Probabilistic finite elements for nonlinear structural dynamics. *Computer Methods in Applied Mechanics and Engineering*, 56:61–81, 1986.

[128] W.A. Lodwick and K.D. Jamison. Special issue: interface between fuzzy set theory and interval analysis. *Fuzzy Sets and Systems*, 135:1–3, 2002.

[129] W.A. Lodwick and K.D. Jamison. Estimating and validating the cumulative distribution of a function of random variables: Toward the development of distribution arithmetic. *Reliable Computing*, 9(2):127–141, 2003.

[130] L. Lucchese, G. Doretto, and G.M. Cortelazzo. A frequency domain technique for range data registration. *IEEE Transactions on Patten Analysis and Machine Intelligence*, 24:1468–1484, 2002.

[131] D.R. Luce and H. Raiffa. *Games and Decisions, Introduction and Critical Survey.* John Wiley & Sons, Inc., New York, 1957.

[132] M. Maceira, S.R. Taylor, C.J. Ammon, X. Yang, and A.A. Velasco. High-resolution rayleigh wave slowness tomography of central asia. *Journal of Geophysical Research*, 110, 2005. paper B06304.

[133] M. Martinez, L. Longpré, V. Kreinovich, S.A. Starks, and H.T. Nguyen. Fast quantum algorithms for handling probabilistic, interval, and fuzzy uncertainty. In *Proceedings of the 22nd International Conference of the North American Fuzzy Information Processing Society NAFIPS'2003*, pages 395–400, Chicago, IL, July 24–26 2003.

[134] S. McWilliam. Anti-optimisation of uncertain structures using interval analysis. *Comput. Struct.*, 79:421–430, 2000.

[135] R.E. Melchers. *Structural Reliability Analysis and Prediction.* John Wiley & Sons, West Sussex, England, second edition, 1999.

[136] R. Michel and J.P. Avouac. Deformation due to the 17 august 1999 izmit, turkey, earthquake measured from spot images. *Journal of Geophysical Research*, 107(B4), 2002.

[137] E.M. Mikhail, J.S. Bethel, and J.M. McGlone. *Introduction to Modern Photogrammetry*. John Wiley & Sons, Inc., New York, 2001.

[138] N. Moës, J. Dolbow, and T. Belytschko. A finite element method for crack growth without remeshing. *Int. J. Numer. Methods Engrg.*, 46:131–150, 1999.

[139] B. Möller, W. Graf, and M. Beer. Fuzzy structural analysis using level-optimization. *Comput. Mech.*, 26(6):547–565, 2000.

[140] R.E. Moore. *Interval Analysis*. Prentice-Hall, Inc., Englewood Cliffs, NJ, 1966.

[141] R.E. Moore. *Methods and Applications of Interval Analysis*. SIAM, Philadelphia, PA, 1979.

[142] R.E. Moore and W.A. Lodwick. Interval analysis and fuzzy set theory. *Fuzzy Sets and Systems*, 135(1):5–9, 2003.

[143] D. Morgenstein and V. Kreinovich. Which algorithms are feasible and which are not depends on the geometry of space-time. *Geombinatorics*, 4(3):80–97, 1995.

[144] R.L. Muhanna and R.L. Mullen. Development of interval based methods for fuzziness in continuum mechanics. In *Proc. ISUMA-NAFIPS'95*, pages 23–45, 1995.

[145] R.L. Muhanna and R.L. Mullen. Formulation of fuzzy finite element methods for mechanics problems. *Compu.-Aided Civ. Infrastruct. Engrg.*, 14:107–117, 1999.

[146] R.L. Muhanna and R.L. Mullen. Uncertainty in mechanics problems-interval-based approach. *J. Engrg. Mech.*, 127(6):557–566, 2001.

[147] R.L. Muhanna, R.L. Mullen, and H. Zhang. Penalty-based solution for the interval finite-element methods. *ASCE, Engineering Mechanics*, 131(10):1102–1111, 2005.

[148] R. Muhanna, V. Kreinovich, P. Šolín, J. Chessa, R. Araiza, and G. Xiang. Interval finite element methods: New directions. In *Proceedings of the Second International*

Workshop on Reliable Engineering Computing, pages 229–243, Savannah, GA, February 22-24 2006.

[149] R.L. Mullen and R.L. Muhanna. Structural analysis with fuzzy-based load uncertainty. In *Proc. 7th ASCE EMD/STD Joint Spec. Conf. on Probabilistic Mech. and Struct. Reliability*, pages 310–313, MA, 1996.

[150] R.L. Mullen and R.L. Muhanna. Bounds of structural response for all possible loadings. *J. Struct. Engrg., ASCE*, 125(1):98–106, 1999.

[151] R.B. Myerson. *Game theory. Analysis of Conflict*. Harvard University Press, Cambridge, MA, 1991.

[152] A. Neumaier. *Interval Methods for Systems of Equations*. Cambridge University Press, Cambridge, 1990.

[153] A. Neumaier and A. Pownuk. Linear systems with large uncertainties, with applications to truss structures. *Reliable Computing*. (to appear).

[154] H.T. Nguyen and V. Kreinovich. Nested intervals and sets: Concepts, relations to fuzzy sets, and applications. In R.B. Kearfott and V. Kreinovich, editors, *Applications of Interval Computations*, pages 245–290. Kluwer, Dordrecht, 1996.

[155] H.T. Nguyen and V. Kreinovich. *Applications of Continuous Mathematics to Computer Science*. Kluwer, Dordrecht, 1997.

[156] H.T. Nguyen and E. A. Walker. *First Course in Fuzzy Logic*. CRC Press, Boca Raton, FL, 1999.

[157] H.T. Nguyen, B. Wu, and V. Kreinovich. Shadows of fuzzy sets – a natural approach towards describing 2-d and multi-d fuzzy uncertainty in linguistic terms. In *Proc. 9th IEEE Int'l Conference on Fuzzy Systems FUZZ-IEEE'2000*, volume 1, pages 340–345, San Antonio, TX, May 7–10 2000.

[158] H.T. Nguyen, T. Wang, and V. Kreinovich. Towards foundations of processing imprecise data: From traditional statistical techniques of processing crisp data to statistical processing of fuzzy data. In Y. Liu, G. Chen, M. Ying, and K.-Y. Cai, editors, *Proceedings of the International Conference on Fuzzy Information Processing: Theories and Applications FIP'2003*, volume II, pages 895–900, Beijing, China, March 1–4 2003.

[159] M.A. Nielsen and I.L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, Cambridge, MA, 2000.

[160] M. Nikravesh. Soft computing-based computational intelligence for reservoir characterization. *Expert Syst. Appl.*, 26(1):19–38, 2004.

[161] P. Nivlet, F. Fournier, and J. Royer. A new methodology to account for uncertainties in 4-d seismic interpretation. In *Proc. 71st Annual Int'l Meeting of Soc. of Exploratory Geophysics SEG'2001*, pages 1644–1647, San Antonio, TX, September 9–14 2001.

[162] P. Nivlet, F. Fournier, and J. Royer. Propagating interval uncertainties in supervised pattern recognition for reservoir characterization. In *Proc. 2001 Society of Petroleum Engineers Annual Conf. SPE'2001*, New Orleans, LA, September 30–October 3 2001. paper SPE-71327.

[163] P. Oliverio. Self-generating pythagorean quadruples and n-tuples. *Fibonacci Quarterly*, pages 98–101, May 1996.

[164] R. Osegueda, V. Kreinovich, L. Potluri, and R. Aló. Non-destructive testing of aerospace structures: Granularity and data mining approach. In *Proc. FUZZ-IEEE'2002*, volume 1, pages 685–689, Honolulu, HI, May 12–17 2002.

[165] R. Osegueda, V. Kreinovich, S. Nazarian, and E. Roldan. Detection of cracks at rivet holes in thin plates using lamb-wave scanning. In T. Kundu, editor, *Smart Nonde-*

structive Evaluation and Health Monitoring of Structural and Biological Systems II, Proc. SPIE, volume 5047, pages 55–66, San Diego, CA, March 3–5 2003.

[166] R. Osegueda, G.R. Keller, S.A. Starks, R. Araiza, D. Bizyaev, and V. Kreinovich. Towards a general methodology for designing sub-noise measurement procedures. In *Proceedings of the 10th IMEKO TC7 International Symposium on Advances of Measurement Science*, volume 1, pages 59–64, St. Petersburg, Russia, June 30–July 2 2004.

[167] C.P. Pantelides and S. Ganzerli. Comparison of fuzzy set and convex model theories in structural design. *Mech. Systems Signal Process.*, 15(3):499–511, 2001.

[168] C.H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Dover Publications, Inc., Mineola, NY, 1998.

[169] R.L. Parker. *Geophysical Inverse Theory*. Princeton University Press, Princeton, NJ, 1994.

[170] M. Paszynski, J. Kurtz, and L. Demkowicz. Parallel, fully automatic *hp*-adaptive 2d finite element package. Technical Report TICAM 04-07, The University of Texas at Austin, 2004.

[171] F. Pavelcik, J. Zelinka, and Z. Otwinowski. Methodology and applications of automatic electron-density map interpretation by six-dimensional rotational and translational serach for molecular fragments. *Acta Crystallografica D*, D58:275–283.

[172] G. Peltzer, F. Crampe, and P. Rosen. The $m_w = 7.1$, hector mine, california earthquake: surface rupture, surface displacement, and fault slip solution from ers sar data. *Earth and Planetary Sciences*, 333:545–555.

[173] E.D. Popova, M. Datcheva, R. Iankov, and T. Schanz. Mechanical models with interval parameters. In G. Gürlebeck, L. Hempel, and C. Könke, editors, *IKM2003:*

Digital Proceedings of 16th International Conference on the Applications of Computer Science and Mathematics in Architecture and Civil Engineering, Weimar, Germany, 2003.

[174] D. Potts, G. Steidl, and M. Tasche. Fast fourier transforms for nonequispaced data: A tutorial. In J.J. Benedetto and P.J.S.G. Ferreira, editors, *Modern Sampling Theory: Mathematics and Applications*, pages 249–274. Birkhaser, Boston, MA, 2001.

[175] D. Potts, G. Steidl, , and A. Nieslony. Fast convolution with radial kernels at nonequispaced knots. *Numer. Math.*, 98(2):329–351, 2004.

[176] W.K. Pratt. *Digital Image Processing*. John Wiley & Sons, New York, 1978.

[177] S. Rabinovich. *Measurement Errors: Theory and Practice*. American Institute of Physics, New York, 1993.

[178] W. Rachowicz, D. Pardo, and L. Demkowicz. Fully automatic *hp*-adaptivity in three dimensions. Technical Report ICES 04-22, The University of Texas at Austin, 2004.

[179] M. Radermacher. Radon transform techniques for alignment and three-dimensional reconstrution from random projections. *Scanning Microscopy*, 11:171–177, 1997.

[180] S.S. Rao and L. Berke. Analysis of uncertain structural systems using interval analysis. *AIAA J.*, 35(4):727–735, 1997.

[181] S.S. Rao and L. Chen. Numerical solution of fuzzy linear equations in engineering analysis. *Int. J. Numer. Meth. Engng.*, 43:391–408, 1998.

[182] B.S. Reddy and B.N. Chatterji. An FFT-based technique for translation, rotation, and scale-invariant image registration. *IEEE Transactions on Image Processing*, 5(8):1266–1271, 1996.

[183] H. Regan, S. Ferson, and D. Berleant. Equivalence of five methods for bounding uncertainty. *Journal of Approximate Reasoning*, 36(1):1–30, 2004.

[184] S. Riazanoff. *SPOT Satellite Geometry Handbook*. SPOT Image, 2002.

[185] D. Robertson, W. Johnston, and W. Nip. Virtual frog dissection: Interactive 3d graphics via the web. In *Proceedings of the Second International WWW Conference '94: Mosaic and the Web*, Chicago, IL, 1994.

[186] N.C. Rowe. Absolute bounds on the mean and standard deviation of transformed data for constant-sign-derivative transformations. *SIAM Journal of Scientific Statistical Computing*, 9:1098–1113, 1988.

[187] S.M. Rump. Solving algebraic problems with high accuracy. In U. Kulisch and W. Miranker, editors, *A New Approach to Scientific Computation*. Academic Press, New York, 1983.

[188] C.G. Schiek. Terrain change detection using aster optical satellite imagery along the kunlun fault, tibet. Master's thesis, The University of Texas at El Paso, Department of Geological Sciences, 2004.

[189] C.G. Schiek, R. Araiza, J.M. Hurtado, A.A. Velasco, V. Kreinovich, and V. Sinyansky. Images with uncertainty: Efficient algorithms for shift, rotation, scaling, and registration, and their applications to geosciences. In M. Nachtegael, D. Van der Weken, and E.E. Kerre, editors, *Soft Computing in Image Processing: Recent Advances*. Springer Verlag, Berlin. (to appear).

[190] G. Schuëller. Computational stochastic mechanics – recent advances. *Computers and Structures*, 79:2225–2234, 2001.

[191] K. Sentz and S. Ferson. Combination of evidence in Dempster-Shafer theory. Technical Report SAND2002-0835, Sandia National Laboratories, 2002.

[192] J.A. Sethian. *Level Set Methods and Fast Marching Methods*. Cambridge University Press, 1999.

[193] G. Shafer. *A Mathematical Theory of Evidence*. Princeton University Press, Princeton, NJ, 1976.

[194] H. Shekarforoush, M. Berthod, and J. Zerubia. Subpixel image registration by estimating the polyphase decomposition of cross power spectrum. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition CVPR'96*, pages 532–537, 1996.

[195] I. Shmulevich and W. Zhang. Binary analysis and optimization-based normalization of gene expression data. *Bioinformatics*, 18(4):555–565, 2002.

[196] C. Bajaj, P. Djeu, V. Siddavanahalli, and A. Thane. Interactive visual exploration of large flexible multi-component molecular complexes. In *Proc. of the Annual IEEE Visualization Conference*, pages 243–250, Austin, TX, October 2004. IEEE Computer Society Press.

[197] A.K. Sinha. *Geoinformatics: Data to Knowledge*. Geological Society of America Publ., Boulder, CO, 2006.

[198] P. Šolín, K. Segeth, and I. Doležel. *Higher-Order Finite Element Methods*. Chapman & Hall/CRC Press, Boca Raton, FL, 2003.

[199] P. Šolín and L. Demkowicz. Goal-oriented *hp*-adaptivity for elliptic problems. *Computer Methods in Applied Mechanics and Engineering*, 193:449–468, 2004.

[200] P. Šolín. *Partial Differential Equations and the Finite Element Method*. J. Wiley & Sons, Hoboken, NJ, 2005.

[201] P. Šolín and T. Vejchodský. On the discrete maximum principle for the *hp*-fem. Technical report, The Universityof Texas at El Paso, Department of Mathematical Science, February 2005. http://www.math.utep.edu/Faculty/solin/new_papers/dmp.pdf; see also http://www.math.utep.edu/Faculty/solin/new_papers/dmp-coll.pdf.

[202] P. Šolín, T. Vejchodský, and R. Araiza. Discrete conservation of nonnegativity for elliptic problems solved by the *hp*-fem. *Mathematics and Computers in Simulation*, 76:205–210, October 2007.

[203] S. Srikrishnan, R. Araiza, H. Xie, S.A. Starks, and V. Kreinovich. Automatic referencing of satellite and radar images. In *Proceedings of the 2001 IEEE Systems, Man, and Cybernetics Conference*, pages 2176–2181, Tucson, AZ, October 7–10 2001.

[204] S. Srikrishnan. Referencing noisy images: complexity of the problem, analysis and efficient implementation of the fast fourier approach. Master's thesis, The University of Texas at El Paso, Department of Computer Science, 2002.

[205] S.A. Starks, S. Nazarian, V. Kreinovich, J. Adidhela, and R. Araiza. Using fft-based data processing techniques to characterize asphaltic concrete mixtures. In *Proceedings of the 11th IEEE Digital Signal Processing Workshop DSP'04*, pages 241–245, Taos Ski Valley, NM, August 1–4 2004.

[206] S.A. Starks, V. Kreinovich, L. Longpré, M. Ceberio, G. Xiang, R. Araiza, J. Beck, R. Kandathi, A. Nayak, and R. Torres. Towards combining probabilistic and interval uncertainty in engineering calculations. In *Proceedings of the Workshop on Reliable Engineering Computing*, pages 193–213, Savannah, GA, September 15–17 2004.

[207] F. Stazi, E. Budyn, J. Chessa, and T. Belytschko. An extended finite element method with higher-order elements for crack problems with curvature. *Computational Mechanics*, 31(1–2):38–48.

[208] G. Steidl. A note on fast fourier transforms for nonequispaced grids. *Advances in Computational Mathematics*, 9:337–352, 1998.

[209] N. Sukumar, D.L. Chopp, N. Moës, and T. Belytschko. Modeling holes and inclusions by level sets in the extended finite element method. *Computer Methods in Applied Mechanics and Engineering*, 190(46–47):6183–6200, 2000.

[210] E. Suli and A. Ware. A spectral method of characteristics for hyperbolic problems. *SIAM J. Numer. Anal.*, 28:423–445, 1991.

[211] P. Suppes, D. M. Krantz, R. D. Luce, and A. Tversky. *Foundations of Measurement. Vol. II. Geometrical, Threshold, and Probabilistic Representations.* Academic Press, San Diego, CA, 1989.

[212] A.R. Thompson, J.M. Moran, and G.W. Swenson Jr. *Interferometry and Synthesis in Radio Astronomy.* Wiley, New York, 2001.

[213] A.N. Tikhonov and V.Y. Arsenin. *Solutions of Ill-Posed Problems.* W.H. Whinston & Sons, Washington, DC, 1977.

[214] R. Tong and R.W. Cox. Rotation of nmr images using the 2d chirp-z transform. *Magnetic Resonance in Medicine*, 41:253–256, 1999.

[215] A. Trautman. Pythagorean spinors and penrose twistors. In S.A. Hugget, L.J. Mason, K.P. Tod, S.T. Tsou, and N.M.J. Woodhouse, editors, *The Geometric Universe; Science, Geometry, and the Work of Roger Penrose.* Oxford Univ. Press, Oxford, 1998.

[216] R. Trejo and V. Kreinovich. Error estimations for indirect measurements. In S. Rajasekaran, P.M. Pardalos, J.H. Reif, and J.D. Rolim, editors, *Handbook on Randomized Computing*, pages 673–729. Kluwer, 2001.

[217] J. Van der Woerd, A.S. Meriaux, Y. Klinger, F.J. Ryerson, Y. Gaudemer, and P. Tapponnier. The 14 november 2001, $m_w = 7.8$ kokoxili earthquake in northern tibet (qinghai province, china). *Seismological Research Letters*, 73(2):125–135, 2002.

[218] N. Van Puymbroeck, R. Michel, R. Binet, J.P. Avouac, and J. Taboury. Measuring earthquakes from optical satellite images. *Applied Optics*, 39(20):3486–3494, 2000.

[219] D.W. Vasco, J.E. Peterson Jr., and E.L. Majer. Resolving seismic anisotropy: Sparse matrix methods for measuring earthquakes from optical satellite images. *Geophysics*, 63(3):970–983, 1998.

[220] S.A. Vavasis. *Nonlinear Optimization: Complexity Issues*. Oxford University Press, New York, 1991.

[221] T. Vejchodský. On the nonnegativity conservation in semidiscrete parabolic problems. In M. Křížek, P. Neittaanmäki, R. Glowinski, and S. Korotov, editors, *Conjugate Gradients Algorithms and Finite Element Methods*, pages 197–210. Springer-Verlag, Berlin, 2004.

[222] T. Vejchodský. Method of lines and conservation of nonnegativity. In *Proc. of the European Congress on Computational Methods in Applied Sciences and Engineering (ECCOMAS 2004)*, Jyväskylä, Finland, 2004.

[223] G.L. Verschuur and K.I. Kellerman. *Galactic and Extragalactic Radio Astronomy*. Springer-Verlag, New York, 1988.

[224] Jr. H.M. Wadsworth, editor. *Handbook of statistical methods for engineers and scientists*. McGraw-Hill Publishing Co., New York, 1990.

[225] P. Walley. *Statistical Reasoning with Imprecise Probabilities*. Chapman & Hall, New York, 1991.

[226] G.W. Walster. Philosophy and practicalities of interval arithmetic. In *Reliability in Computing*, pages 309–323. Academic Press, New York, 1988.

[227] G.W. Walster and V. Kreinovich. For unknown-but-bounded errors, interval estimates are often better than averaging. *ACM SIGNUM Newsletter*, 31(2):6–19, 1996.

[228] A.F. Ware. Fast approximate fourier transforms for irregularly spaced data. *SIAM Rev.*, 40(4):838–856.

[229] R. Williamson and T. Downs. Probabilistic arithmetic i: numerical methods for calculating convolutions and dependency bounds. *International Journal of Approximate Reasoning*, 4:89–158, 1990.

[230] B. Wu, H.T. Nguyen, and V. Kreinovich. Real-time algorithms for statistical analysis of interval data. In *Proceedings of the International Conference on Information Technology InTech'03*, pages 483–490, Chiang Mai, Thailand, December 17–19 2003.

[231] G. Xiang, S.A. Starks, V. Kreinovich, and L. Longpré. New algorithms for statistical analysis of interval data. In *Proceedings of the Workshop on State-of-the-Art in Scientific Computing PARA'04*, volume 1, pages 123–129, Lyngby, Denmark, June 20–23 2004.

[232] G. Xiang. Fast algorithm for computing the upper endpoint of sample variance for interval data: case of sufficiently accurate measurements. *Reliable Computing*. (in press).

[233] H. Xie, N. Hicks, G.R. Keller, H. Huang, and V. Kreinovich. Automatic image registration based on a fft algorithm and idl/envi. In *Proceedings of the ICORG-2000 International Conference on Remote Sensing and GIS/GPS*, volume 1, pages 397–402, Hyderabad, India, December 1–4 2000.

[234] H. Xie, N. Hicks, G.R. Keller, H. Huang, and V. Kreinovich. Implementation, test, and analysis of an automatic image registration algorithm based on FFT and IDL/ENVI. *Computers and Geosciences*, 29(8):1045–1055, 2003.

[235] J.F. Young, S. Mindess, R.J. Gray, and A. Bentur. *Science and Technology of Civil Engineering Materials*. Prentice Hall, Englewood Cliffs, NJ, 1997.

[236] J. Yu, E. Cowgill, and S. Healy. *Digital Stereomapping in Imagine Orthobase*, pages 1–10. California Institute of Technology, 2003.

[237] L.A. Zadeh. Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets and Systems*, 1:3–28, 1978.

[238] C.A. Zelt and P.J. Barton. Three-dimensional seismic refraction tomography: A comparison of two methods applied to data from the faeroe basin. *J. Geophysical Research*, 103(B4):7187–7210, 1998.

[239] W. Zhang, I. Shmulevich, and J. Astola. *Microarray Quality Control*. Wiley, Hoboken, NJ, 2004.

[240] B. Zitová and J. Flusser. Image registration methods: a survey. *Image and Vision Computing*, 21:977–1000, 2003.

Curriculum Vitae

Roberto Araiza Landeros was born on September 10, 1979, in Cd. Juárez, México, the youngest son of Roberto Araiza Porras and Bertha Alicia Landeros Trevizo. He entered the University of Texas at El Paso in the Fall of 1996 immediately after graduating from "El Chamizal" High School in Cd. Juárez. While pursuing his Bachelor's degree in Computer Science, he worked first as a Math Tutor and then as a Research and Teaching Assistant in the Department of Computer Science. He received his Bachelor's degree in Computer Science in December of 2000, graduating Summa Cum Laude and with a University Honors Degree. He was also selected as one of the Top Ten Seniors for the year 2000.

In the Spring of 2001, Roberto entered the Graduate School of the University of Texas at El Paso. While pursuing his Master's degree in Computer Science he worked as a Graduate Research and Teaching Assistant. He also worked for IBM Corporation in Austin, Texas, as a Software Intern the summers of 2001, 2002, and 2003. He received his Master's degree in Computer Science in December of 2003.

In the Spring of 2004, Roberto began his doctoral program at the University of Texas at El Paso. While pursuing his Ph.D. degree in Computer Science, with an emphasis on computational science, he also worked as a Graduate Research Assistant in the area of systems analysis. He will complete his studies in December of 2007.

Roberto has presented his research at several conferences in venues such as Paris, France; San Diego, California; Albuquerque, New Mexico; and Tucson, Arizona. He published 18 papers, most of them in international journals and proceedings of international conferences.

He is a member of the National Honor Society for the Computing Sciences Upsilon Pi Epsilon, the UTEP Chapter of the Association for Computing Machinery, and the Center for Theoretical Research and its Applications in Computer Science (TRACS).

Permanent address: Blvd. Tomás Fernández #6935
Cd. Juárez, Chihuahua, México C.P. 32410